

Université  
de Toulouse

# THÈSE

## En vue de l'obtention du DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

**Délivré par :**

Université Toulouse III Paul Sabatier (UT3 Paul Sabatier)

**Discipline ou spécialité :**

Informatique

---

**Présentée et soutenue par :**

Arlind Kopliku

**le :** mercredi 7 décembre 2011

**Titre :**

Approaches to implement and evaluate aggregated search  
(Approches pour implémenter et évaluer la recherche d'information agrégée)

---

**Ecole doctorale :**

Mathématiques Informatique Télécommunications (MITT)

**Unité de recherche :**

SIG

**Directeur(s) de Thèse :**

Mohand Boughanem

**Rapporteurs :**

Mounia LALMAS  
Eric GAUSSIER

Professeur, Université de Glasgow  
Professeur, Université de Grenoble I

**Membre(s) du jury :**

Karen Pinel -Sauvagnat,	Maître de conférences, Université de Toulouse III,	co-encadrante
Claude Chrisment,	Professeur, Université de Toulouse III,	examineur
Patrick Gallinari,	Professeur, Université de Paris VI,	examineur



# Approaches to implement and evaluate aggregated search

Arlind Kopliku

Institut de Recherche en Informatique de Toulouse, UMR 5505 CNRS,  
SIG-RFI  
Université Paul Sabatier 118 route de Narbonne F-31062 Toulouse Cedex 9

Phd:

Arlind Kopliku .....

Advisor:

Karen Pinel-Sauvagnat .....

Supervisor:

Mohand Boughanem .....

Comments, corrections, and other feedback most welcome at:

kopliku@irit.fr, arlind29@gmail.com, bougha@irit.fr, sauvagna@irit.fr

## Remerciements

A la clôture de ces années de doctorat, j'ai envie d'attacher à ce rapport une page ou deux de phrases descriptives et de sentiments qu'on ne peut pas détacher de la thèse, car ces éléments en font partie. J'ai vécu une partie de ma thèse comme un défi et enfin je me suis rendu compte que c'est une des plus intense, sincère et complète formation, accompagnée de moments de vie, de sentiments, d'amis et ami-collègues. Je vais pourtant graver dans les paragraphes qui suivent les remerciements, les sentiments vis-à-vis des gens qui ont contribué directement ou indirectement à cette expérience de vie.

Tout d'abord je veux remercier les personnes qui ont eu la plus grande influence dans ces trois années de thèse, les personnes qui ont pris un étudiant de master et l'ont amené jusqu'au but avec professionnalisme. C'est à mon directeur de thèse **Mohand Boughanem** et ma co-encadrante **Karen Pinel-Sauvagnat** que va en premier ma gratitude.

C'est grâce à Mohand Boughanem que j'ai eu l'occasion de faire cette thèse en recherche d'information, domaine qui m'intriguait et qui m'intrigue toujours. C'est sans doute une chance d'être dirigé par un des meilleurs et un des plus actifs de ce domaine. Cette thèse doit beaucoup à son recul, ses conseils, son expertise et à son support. J'ai apprécié son rôle de guide, son sens de l'humour et son enthousiasme pour la recherche et la vie.

Parallèlement, Karen m'a donné tous ce que de mieux peut donner une co-encadrante. Elle m'a accompagné pendant toutes les étapes de la thèse sans jamais me nier son aide. Depuis le début, je savais qu'il y avait un bureau où je pouvais frapper en cas de besoin et j'y trouvais toujours les réponses qui me manquaient. Je lui serai toujours reconnaissant pour sa sincérité, rigueur et surtout son support professionnel et amical.

Je tiens à remercier les rapporteurs **Mounia Lalmas** et **Eric Gaussier** ainsi que les examinateurs **Claude Chrisment** et **Patrick Gallinari** qui m'ont fait l'honneur de faire partie de mon jury de thèse et de s'intéresser à mon travail. Je suis reconnaissant de leur retour précis et professionnel sur le manuscrit.

Je tiens à remercier Cécile Paris, Paul Thomas et Stephen Wan chercheurs estimés Australiens qui m'ont fait le plaisir de m'accueillir au CSIRO ICT Centre à Sydney. Si ces deux mois m'ont apporté un enrichissement professionnel et personnel, c'est grâce à eux. J'en garderai un très bon souvenir.

Je tiens à remercier aussi mes amis quotidiens du laboratoire. Nous avons rigolé ensemble tellement que ces remerciements ne sembleront pas sincères. Dans les moments de stress et de difficulté, ils étaient là. Let the

list begin: Imen, Hamdi, Lamjed, Cyril, Deniz, Firas, Ines, Dana, Madalina, Faten. Nous avons passé au moins une heure (parfois deux, ...) par jour ensemble entre repas et pauses. Il faut dire que c'est aussi grâce aux pauses qu'on avance une thèse. Pendant ces années, le bureau 401 à eu beaucoup de visiteurs et en dehors du bruit il y reste l'amitié et le respect. Je ne peux pas oublier ceux qui étaient là avant cette année tel que Guillaume, Fatma, Malik, Karim, Moulazem. Si je commence avec la liste d'amis de Toulouse (hors équipe et laboratoire), je risque de ne plus m'arrêter et il faudrait classer par nationalité. En tous cas, merci Céline, Victor, Candice, Emeline, Emilie, Etienne, Diego, Irene, Federico, Mattia, Stéphanie, Bruno, Marco, Corina, Loveday, Giulia, Alice, Marta, Vincezo, Eleonora, Luisa, Nina ... Rien de mieux que de finir par Rodrigo, ami et colocataire sincère.

Je veux aussi exprimer le plaisir de travailler et collaborer avec les différents membres de l'équipe SIG ou Pyramide tant pour les enseignements (Lynda, Olivier, Wahiba, Frank ...), que pour les conseils (Guillaume, Ronan, ...). Je remercie Firas et Ines pour leur contribution qui est fondamentale dans deux des chapitres de cette thèse.

Enfin, je ne peux pas oublier mes parents et ma sœur que je devrais remercier tous les jours, mais je n'en ferai jamais assez et souvent c'est un "merci" silencieux.

# Abstract

*Aggregated search* or *aggregated retrieval* can be seen as a third paradigm for information retrieval following the *boolean retrieval* paradigm and the *ranked retrieval* paradigm. In the first two, we are returned respectively sets and ranked lists of search results. It is up to the time-poor user to scroll this set/list, scan within different documents and assemble his/her information need. Alternatively, aggregated search not only aims the identification of relevant information nuggets, but also the assembly of these nuggets into a coherent answer.

In this work, we present at first an analysis of related work to aggregated search which is analyzed with a general framework composed of three steps: *query dispatching*, *nugget retrieval* and *result aggregation*. Existing work is listed aside different related domains such as relational search, federated search, question answering, natural language generation, etc. Within the possible research directions, we have then focused on two directions we believe promise the most namely : *relational aggregated search* and *cross-vertical aggregated search*.

- *Relational aggregated search* targets relevant information, but also relations between relevant information nuggets which are to be used to assemble reasonably the final answer. In particular, there are three types of queries which would easily benefit from this paradigm: *attribute queries* (e.g. president of France, GDP of Italy, major of Glasgow, ...), *instance queries* (e.g. France, Italy, Glasgow, Nokia e72, ...) and *class queries* (countries, French cities, Nokia mobile phones, ...). We call these queries as *relational queries* and we tackle with three important problems concerning the information retrieval and aggregation for these types of queries.

First, we propose an attribute retrieval approach after arguing that attribute retrieval is one of the crucial problems to be solved. Our approach relies on the HTML tables in the Web. It is capable to identify useful and relevant tables which are used to extract relevant attributes for whatever queries. The different experimental results show that our approach is effective, it can answer many queries with high coverage and it outperforms state of the art techniques.

Second, we deal with result aggregation where we are given relevant instances and attributes for a given query. The problem is particularly interesting for class queries where the final answer will be a table

with many instances and attributes. To guarantee the quality of the aggregated result, we propose the use of different weights on instances and attributes to promote the most representative and important ones. The third problem we deal with concerns instances of the same class (e.g. France, Germany, Italy, ... are all instances of the same class). Here, we propose an approach that can massively extract instances of the same class from HTML lists in the Web. All proposed approaches are applicable at Web-scale and they can play an important role for relational aggregated search.

Finally, we propose 4 different prototype applications for relational aggregated search. They can answer different types of queries with relevant and relational information. Precisely, we not only retrieve attributes and their values, but also passages and images which are assembled into a final focused answer. An example is the query “Nokia e72” which will be answered with attributes (e.g. price, weight, battery life, ...), passages (e.g. description, reviews, ...) and images. Results are encouraging and they illustrate the utility of relational aggregated search.

- The second research direction that we pursued concerns *cross-vertical aggregated search*, which consists of assembling results from different vertical search engines (e.g. image search, video search, traditional Web search, ...) into one single interface. Here, different approaches exist in both research and industry. Our contribution concerns mostly evaluation and the interest (advantages) of this paradigm. We propose 4 different studies which simulate different search situations. Each study is tested with 100 different queries and 9 vertical sources. Results are interesting. We could clearly identify new advantages of this paradigm and we could identify different issues with evaluation setups. In particular, we observe that traditional information retrieval evaluation is not the fastest but it remains the most realistic.

To conclude, we propose different studies with respect to two promising research directions. On one hand, we deal with three important problems of relational aggregated search following with real prototype applications with encouraging results. On the other hand, we have investigated on the interest and evaluation of cross-vertical aggregated search. Here, we could clearly identify some of the advantages and evaluation issues. In a long term perspective, we foresee a possible combination of these two kinds of approaches to provide relational and cross-vertical information retrieval incorporating more focus, structure and multimedia in search results.



# Résumé

La *recherche d'information agrégée* peut être vue comme un troisième paradigme de recherche d'information après la *recherche d'information ordonnée* (ranked retrieval) et la *recherche d'information booléenne* (boolean retrieval). Les deux paradigmes les plus explorés jusqu'à aujourd'hui retournent un ensemble ou une liste ordonnée de résultats. C'est à l'utilisateur de parcourir ces ensembles/listes et d'en extraire l'information nécessaire qui peut se retrouver dans plusieurs documents. De manière alternative, la recherche d'information agrégée ne s'intéresse pas seulement à l'identification des granules (nuggets) d'information pertinents, mais aussi à l'assemblage d'une réponse agrégée contenant plusieurs éléments.

Dans nos travaux, nous analysons les travaux liés à la recherche d'information agrégée selon un schéma général qui comprend 3 parties: *dispatching de la requête*, *recherche de granules d'information* et *agrégation du résultat*. Les approches existantes sont groupées autour de plusieurs perspectives générales telle que la recherche relationnelle, la recherche fédérée, la génération automatique de texte, etc. Ensuite, nous nous sommes focalisés sur deux pistes de recherche selon nous les plus prometteuses: (i) la *recherche agrégée relationnelle* et (ii) la *recherche agrégée inter-verticale*.

- La *recherche agrégée relationnelle* s'intéresse aux relations entre les granules d'information pertinents qui servent à assembler la réponse agrégée. En particulier, nous nous sommes intéressés à trois types de requêtes notamment: *requête attribut* (ex. président de la France, PIB de l'Italie, maire de Glasgow, ...), *requête instance* (ex. France, Italie, Glasgow, Nokia e72, ...) et *requête classe* (pays, ville française, portable Nokia, ...). Pour ces requêtes qu'on appelle *requêtes relationnelles* nous avons proposés trois approches pour permettre la recherche de relations et l'assemblage des résultats.

Nous avons d'abord mis l'accent sur la recherche d'attributs qui peut aider à répondre aux trois types de requêtes. Nous proposons une approche à large échelle capable de répondre à des nombreuses requêtes indépendamment de la classe d'appartenance. Cette approche permet l'extraction des attributs à partir des tables HTML en tenant compte de la qualité des tables et de la pertinence des attributs. Les différentes évaluations de performances effectuées prouvent son efficacité qui dépasse les méthodes de l'état de l'art.

Deuxièmement, nous avons traité l'agrégation des résultats composés d'instances et d'attributs. Ce problème est intéressant pour répondre

à des requêtes de type classe avec une table contenant des instances (lignes) et des attributs (colonnes). Pour garantir la qualité du résultat, nous proposons des pondérations sur les instances et les attributs promouvant ainsi les plus représentatifs. Le troisième problème traité concerne les instances de la même classe (ex. France, Italie, Allemagne, ...). Nous proposons une approche capable d'identifier massivement ces instances en exploitant les listes HTML. Toutes les approches proposées fonctionnent à l'échelle Web et sont importantes et complémentaires pour la recherche agrégée relationnelle.

Enfin, nous proposons 4 prototypes d'application de recherche agrégée relationnelle. Ces derniers peuvent répondre des types de requêtes différents avec des résultats relationnels. Plus précisément, ils recherchent et assemblent des attributs, des instances, mais aussi des passages et des images dans des résultats agrégés. Un exemple est la requête "Nokia e72" dont la réponse sera composée d'attributs (ex. prix, poids, autonomie batterie, ...), de passages (ex. description, reviews, ...) et d'images. Les résultats sont encourageants et illustrent l'utilité de la recherche agrégée relationnelle.

- La *recherche agrégée inter-verticale* s'appuie sur plusieurs moteurs de recherche dits verticaux tel que la recherche d'image, recherche vidéo, recherche Web traditionnelle, etc. Son but principal est d'assembler des résultats provenant de toutes ces sources dans une même interface pour répondre aux besoins des utilisateurs. Les moteurs de recherche majeurs et la communauté scientifique nous offrent déjà une série d'approches. Notre contribution consiste en une étude sur l'évaluation et les avantages de ce paradigme. Plus précisément, nous comparons 4 types d'études qui simulent des situations de recherche sur un total de 100 requêtes et 9 sources différentes. Avec cette étude, nous avons identifiés clairement des avantages de la recherche agrégée inter-verticale et nous avons pu déduire de nombreux enjeux sur son évaluation. En particulier, l'évaluation traditionnelle utilisée en RI, certes la moins rapide, reste la plus réaliste.

Pour conclure, nous avons proposé des différentes approches et études sur deux pistes prometteuses de recherche dans le cadre de la recherche d'information agrégée. D'une côté, nous avons traité trois problèmes importants de la recherche agrégée relationnelle qui ont porté à la construction de 4 prototypes d'application avec des résultats encourageants. De l'autre côté, nous avons mis en place 4 études sur l'intérêt et l'évaluation de la recherche agrégée inter-verticale qui ont permis d'identifier les enjeux d'évaluation et les avantages du paradigme. Comme suite à long terme de ce travail, nous pouvons envisager une recherche d'information qui intègre plus de granules relationnels et plus de multimédia.

# Contents

<b>I</b>	<b>Part 1: Introduction</b>	<b>1</b>
<b>1</b>	<b>Context, contribution and layout</b>	<b>3</b>
1.1	Context . . . . .	3
1.1.1	Relational aggregated search . . . . .	5
1.1.2	Cross-vertical aggregated search . . . . .	6
1.2	Contribution . . . . .	7
1.2.1	Relational aggregated search . . . . .	7
1.2.2	Cross-vertical aggregated search . . . . .	8
1.3	Thesis outline . . . . .	9
<b>II</b>	<b>Part 2: Aggregated search</b>	<b>11</b>
<b>2</b>	<b>Boolean, ranked and aggregated IR</b>	<b>13</b>
2.1	Information Retrieval Process . . . . .	13
2.1.1	Indexing . . . . .	14
2.1.2	Querying: Information need and queries . . . . .	17
2.1.3	Query matching . . . . .	17
2.2	Boolean and ranked retrieval . . . . .	18
2.2.1	Boolean retrieval . . . . .	18
2.2.2	Ranked retrieval . . . . .	19
2.2.3	Limits of ranked retrieval . . . . .	19
2.3	Conclusion: Towards aggregated retrieval . . . . .	20
<b>3</b>	<b>Aggregated search</b>	<b>23</b>
3.1	Introduction . . . . .	23
3.2	Issues . . . . .	25
3.3	A generic framework for aggregated search . . . . .	26
3.3.1	Query dispatching . . . . .	27
3.3.2	Nuggets retrieval . . . . .	28
3.3.3	Result aggregation . . . . .	29
3.4	Different perspectives . . . . .	31
3.4.1	Question Answering . . . . .	32

3.4.2	Natural Language Generation . . . . .	33
3.4.3	Relational aggregated search . . . . .	34
3.4.4	Federated search . . . . .	36
3.4.5	Meta-search . . . . .	37
3.4.6	Data fusion . . . . .	38
3.4.7	Mash-up . . . . .	38
3.4.8	Cross-vertical aggregated search . . . . .	39
3.4.9	Domain-specific applications . . . . .	40
3.5	Conclusions . . . . .	43
<b>4</b>	<b>Relational aggregated search</b>	<b>45</b>
4.1	Introduction . . . . .	45
4.2	Framework and issues . . . . .	46
4.3	Relational queries . . . . .	49
4.4	How to acquire relations? . . . . .	50
4.4.1	Instance-class relation . . . . .	52
4.4.2	Instance-instance relation . . . . .	53
4.4.3	Instance-attribute relation . . . . .	55
4.4.4	Instance-nugget relations and nugget-nugget relations	56
4.5	How to retrieve relations? . . . . .	56
4.6	Result aggregation . . . . .	57
4.7	Case studies . . . . .	58
4.7.1	Object-level search . . . . .	59
4.7.2	Opinion mining . . . . .	60
4.8	Conclusions . . . . .	61
<b>5</b>	<b>Cross-vertical aggregated search</b>	<b>63</b>
5.1	Introduction . . . . .	63
5.2	Advantages of cross-vertical aggregated search . . . . .	65
5.3	Source selection and representation . . . . .	66
5.4	Result aggregation . . . . .	68
5.5	Result presentation . . . . .	70
5.6	Evaluation . . . . .	74
5.7	Conclusions . . . . .	75
<b>III</b>	<b>Part 3: Relational aggregated search: Attribute retrieval, result aggregation, instance lists and applications</b>	<b>77</b>
<b>6</b>	<b>Attribute retrieval</b>	<b>79</b>
6.1	Introduction . . . . .	79
6.2	Attribute retrieval . . . . .	80
6.2.1	Filters . . . . .	82
6.2.2	Ranking attributes by relevance . . . . .	84

6.3	Experimental setup . . . . .	87
6.3.1	Filtering setup and evaluation . . . . .	87
6.3.2	Attribute retrieval evaluation . . . . .	87
6.4	Results . . . . .	88
6.4.1	Performance of filtering . . . . .	88
6.4.2	Impact of the relevance features on attribute retrieval	90
6.4.3	Impact of filters on attribute retrieval . . . . .	90
6.4.4	Performance by search situation . . . . .	91
6.4.5	Estimating recall . . . . .	94
6.4.6	Comparison with state of the art . . . . .	94
6.5	Conclusions . . . . .	95
<b>7</b>	<b>Result aggregation: Building useful tabular results</b>	<b>97</b>
7.1	Introduction . . . . .	97
7.2	Result aggregation approach . . . . .	99
7.2.1	Instance selection . . . . .	99
7.2.2	Attribute ranking . . . . .	101
7.3	Experimental setup . . . . .	102
7.3.1	Dataset . . . . .	102
7.3.2	Tables . . . . .	102
7.3.3	Evaluation . . . . .	103
7.4	Results . . . . .	104
7.5	Conclusions . . . . .	106
<b>8</b>	<b>Lists of instances and set expansion</b>	<b>107</b>
8.1	Introduction . . . . .	107
8.2	Mining for siblings' list . . . . .	108
8.2.1	Parsing and filtering . . . . .	109
8.2.2	Mining and extracting . . . . .	110
8.3	Experimental setup . . . . .	111
8.4	Experimental results . . . . .	113
8.4.1	Distribution of siblings' sets . . . . .	113
8.4.2	Mining by feature . . . . .	113
8.4.3	Extracting siblings' lists through classification . . . . .	115
8.5	Conclusions . . . . .	117
<b>9</b>	<b>Prototypes</b>	<b>119</b>
9.1	Introduction . . . . .	119
9.2	Querying . . . . .	120
9.3	More retrieval: attribute values, passages and images . . . . .	122
9.3.1	Attribute value retrieval . . . . .	122
9.3.2	Passage retrieval . . . . .	123
9.3.3	Image retrieval . . . . .	123
9.4	Result aggregation and prototypes . . . . .	123

9.4.1	Prototype “ <i>valoir</i> ” . . . . .	124
9.4.2	Prototype “ <i>pouvoir</i> ” . . . . .	125
9.4.3	Prototype “ <i>revoir</i> ” . . . . .	125
9.4.4	Prototype “ <i>savoir</i> ” . . . . .	126
9.5	Conclusions . . . . .	129
<b>IV</b>	<b>Part 4: Cross-vertical aggregated search: Interest and evaluation</b>	<b>131</b>
<b>10</b>	<b>Evaluation of cvAS</b>	<b>133</b>
10.1	Introduction . . . . .	133
10.2	Experimental setup . . . . .	134
10.2.1	Definition of source relevance . . . . .	135
10.2.2	Queries . . . . .	135
10.2.3	Sources and participants . . . . .	135
10.2.4	Evaluation interface . . . . .	136
10.2.5	Tasks description . . . . .	137
10.2.6	Task deployment . . . . .	138
10.3	Results . . . . .	139
10.3.1	Notation . . . . .	139
10.3.2	Interest of cross-vertical aggregated search . . . . .	139
10.3.3	Complementary sources? . . . . .	141
10.3.4	Impact of relevance and query types on evaluation . . . . .	143
10.3.5	Questionnaires . . . . .	145
10.4	Discussion . . . . .	146
10.5	Conclusions . . . . .	147
<b>V</b>	<b>Part V: Conclusions and future work</b>	<b>149</b>
<b>11</b>	<b>Conclusions and future work</b>	<b>151</b>
11.1	Conclusions . . . . .	151
11.1.1	Relational aggregated search . . . . .	151
11.1.2	Cross-vertical aggregated search . . . . .	153
11.2	Future work . . . . .	154
11.2.1	Relational aggregated search . . . . .	154
11.2.2	Cross-vertical aggregated search . . . . .	155
<b>A</b>	<b>Datasets</b>	<b>157</b>
A.1	Dataset for attribute retrieval evaluation . . . . .	157
A.2	Dataset for the cross-vertical aggregated search study . . . . .	159
	<b>Bibliographic references</b>	<b>161</b>

## Part I

### Part 1: Introduction





# Chapter 1

## Context, contribution and layout

### 1.1 Context

Most of current Information Retrieval (IR) systems are instances of the ranked retrieval paradigm i.e. in response to a query they return a list of ranked documents that match the query. In general, the user looking for relevant material browses and examines the returned documents to find those that are likely to fulfill his need. If lucky, the user will find in this list the document that satisfies completely his need. However, it is often the case when one document alone is not enough i.e. the relevant information is scattered in different documents. In this case, the user should collect and aggregate the pieces of information coming from different documents to build the most appropriate response to his need. The continuous advancement in Information Retrieval turns the ranked retrieval into a questionable paradigm. Today, it is possible to retrieve entire documents as well as parts of documents. It is possible to retrieve textual documents as well as multimedia documents. Let an information nugget be a generalization of content of some granularity and multimedia format. Within the new research questions that come up we can forward the following. Can these different information nuggets be combined differently to increase IR system utility? Can we provide more focused and coherent results? Is the ranked list the best way to show search results? The alternative can be *aggregated search* (*aggregated retrieval*) which corresponds to a new paradigm where search results should not only be ranked but also assembled with each other.

The key problem which makes the aggregated search complex comes mainly from the fact that the assembly of information is topic-dependent i.e. aggregated results cannot be built a priori. Indeed, one cannot predict and construct all possible combinations that can potentially answer the user queries. The composition of information nuggets that meet the query con-

straints is made upon the query. This leads to several research questions related to the identification of candidate information nuggets for aggregation, the definition of the properties and the theoretical framework that may support the evaluation of the quality of an aggregated result.

The most well-known instances of aggregated search are met in the major Web search engines. Since few, these search engines do not return uniform lists of Web pages. They also include results of different type such as images, news, videos, definitions, etc. The query “define brontosaurus” issued to Google<sup>1</sup> is indeed answered with a definition, followed by a list of Web pages. The query “brontosaurus” alone issued to Google<sup>2</sup> is answered with a Wikipedia article, images and other Web pages. We can see that Web search provides more focus and diversity in its results. To do so, major search engines combine results from traditional Web search (Web page retrieval) and vertical search engines (video search, image search, ...). We refer to this approach as *cross-vertical aggregated search (cvAS)*.

Current Web search represents initial effort towards aggregated search, but they do not perform any explicit “assembly” of information. Documents are just represented as bags of words and they are matched uniquely to the user query. Consequently, retrieved search results are just a list of unrelated items sometimes difficult to explore. It remains up to the user to select the information he needs and to assemble it. Another form of aggregated search that we define as *relational aggregated search (RAS)* relies on the hypotheses that information can often be decomposed into smaller information nuggets and information nuggets can be put in relation with each other. In literature [35, 37], the term *relational search* is used to refer to Information Retrieval based on classes (e.g. countries, cities, architects, ...), instances (France, London, ...) and attributes (capital, GDP, ...). We generalize this term to *relational aggregated search (RAS)* to refer to Information Retrieval based on information nuggets and their relations.

From our perspective, cross-vertical aggregated search and relational aggregated search are the most promising instances of aggregated search. Nevertheless, aggregated search refers to a broad paradigm which can be instantiated in different forms. We could identify related approaches in different areas including question answering, natural language generation, focused retrieval, object-level search, etc.

In this chapter, we introduce our work. We start with background knowledge on relational aggregated search (section 1.1.1) and cross-vertical aggregated search (section 1.1.2). Then we provide an introduction to our contribution. In the end of this chapter, we describe the thesis layout.

---

<sup>1</sup>August 2011

<sup>2</sup>August 2011

### 1.1.1 Relational aggregated search

Relational aggregated search is a simple paradigm which fits well within aggregated search. It relies on the hypotheses that information can be split into information nuggets and the latter can be related to each other. From this perspective, the relational aggregated search should retrieve information nuggets and their relations, which are to be used to coherently assemble the final search result. Here, we distinguish some immediate research questions that need to be answered : *(i)* how can we extract and relate information ? *(ii)* which queries benefit more from this approach ? *(iii)* how should results be aggregated ? We will briefly introduce each of these issues.

In this context, we distinguish three important extracts of information namely *classes* (e.g. countries, French wines, US presidents), *instances* (e.g. France, USA, Italy) and *attributes* (e.g. capital:Paris, area:674,843 km<sup>2</sup>, demonym:French ...). For these extracts, relations are common and often implicit e.g. “capital:Paris” *is an attribute of the instance* “France” or “France” *is an instance of the class* “countries”. These information extracts can play a crucial role in relational aggregated search, because they are common in information and they have semantic relations with each other. However, there are many other relations that fit in this framework. Identifying the ones that enables better search is one of the main issues in relational aggregated search.

In order to clearly understand the utility of relational aggregated search, we need to identify beforehand the queries that will benefit the most from this approach. For illustration, we forward here 3 types of queries which can be answered through this paradigm with clear benefits:

- i.* attribute query (“GDP of UK”, “address of Hotel Bellagio”)
- ii.* instance query (“Samsung Galaxy S”, “Scotland”, “Oscar Wilde”)
- iii.* class query (“Toshiba notebooks”, “British writers”)

Each of these queries can be answered differently. The result aggregation process should be adapted consequently. When the query is specifically asking for an attribute *(i)*, the best choice can be returning its value right away. When the query is an instance *(ii)*, the best choice can be a summary of salient attributes (properties). When the query is a class of instances *(iii)*, the result can be a comparative table of the class instances with their attributes (names and values). The above are just some result aggregation examples inspired from existing work [35].

Until now, relational search is enabled by information extraction techniques [9] and mining within semi-structured data [36]. Existing techniques can discover many information extracts and their relations. Nevertheless, their use for information retrieval remains limited. Our goal in this work is

to increase the recall of relational aggregated search in both terms of queries that can be answered and relations and content that can be retrieved.

### 1.1.2 Cross-vertical aggregated search

Cross-vertical aggregated search is an example of aggregated search with multiple sources. The sources include the traditional Web search engine that retrieves Web pages, but also the vertical search engines (image search, video search, news search, ...). Search with multiple sources has already been addressed in federated search, meta-search, data fusion, etc. Still, cross-vertical aggregated search represents a new research direction, because it relies on heterogeneous sources with determined specialities. This has several advantages: it adds more focus and diversity within search results; it allows user to query multiple sources from one interface; it provides visibility to vertical search engines, and so on [154, 123].

Research in cross-vertical aggregated search has taken three main directions: *(i)* source (vertical) selection, *(ii)* result aggregation and presentation *(iii)* interest and evaluation. We will detail briefly each of them below.

Querying multiple sources can introduce an important delay in query answering time. Though, many of the existing approaches perform what is known as source (vertical) selection to determine which sources are likely to be useful. To do so, it is common to have a central representation of each source, which can be quickly matched with the query. This representation can be built based on query logs, samples of documents, etc. [14].

The retrieved results from multiple sources can be assembled in different ways. Among these, there are two broad categories namely blended and unblended result aggregation [166]. The unblended approach consists in keeping results from different sources in different panels. The blended approach consists in ranking in the same panel results from all sources. The latter approach has taken the lead in major Web search engines. Ranking results with each other is not easy. The relevance scores from different sources are not directly comparable, though new scoring functions are needed [12, 139].

Last but not least, evaluation of cross-vertical aggregated search remains an open research question. Until now, different evaluation methodologies have been undertaken for evaluating the effectiveness of cross-vertical aggregated search. They tackle different issues such as source (vertical) selection [14, 104, 108], result ranking [13] and visualization interfaces [166, 168, 173]. Although there is an increasing interest, evaluation remains an open problem, because until now there is no common agreement on evaluation measures and possible evaluation benchmarks. As well, it is not clear whether we can generalize existing techniques to work well across heterogeneous approaches. In this work, we focus mainly on the interest and evaluation of cross-vertical aggregated search.

## 1.2 Contribution

This thesis presents a long journey within aggregated search. In particular, we focus on two research directions we believe more promising concretely relational aggregated search and cross-vertical aggregated search. Though, our contribution forks in two parts, one per research direction. In the next two sections, we present our contribution respectively for relational aggregated search following and cross-vertical aggregated search.

### 1.2.1 Relational aggregated search

We define and formalize for the first time (at our knowledge) relational aggregated search as an instance of aggregated search. To do so, we rely on previous work in Information Extraction, object-level search and entity-oriented search. Research in these directions in conjunction with information Retrieval provides enough fluidity to envisage more focus and aggregation for information retrieval. Our contribution accounts for the definition of a relational framework, various approaches and research investigation. We summarize our contribution through some key points and brief description:

- Instead of extracting offline information and relations, we present approaches for online retrieval of relations and information [93, 96, 97]. In particular, we focus on attribute retrieval a novel and challenging problem which allows answering many queries in the context of relational aggregated search. This approach allows relating instances and classes to their attributes. In fact, we do not extract attributes through binary “yes”/“no” classification, but we perform topical (query based) retrieval where we score each candidate relation with respect to relevance scores. Concretely, we propose a recall-oriented approach to retrieve attributes from HTML tables in the Web (probably the largest source of relational data). Given an instance as a query, we first identify candidate relevant tables. Then, we apply 3 filters that tell respectively: *(i)* is the table relational, *(ii)* has the table a header, *(iii)* the conformity of its attributes and values. Then, we rank candidate attributes with a combination of relevance features. This approach is tested for three different situations. First, we retrieve relevant attributes for one given instance (e.g. “University of Strathclyde”). Second, we retrieve attributes for a given class (e.g. “universities”) represented as a set of instances. Third, we retrieve attributes for one instance when some other similar instances are given (from the same class).
- After attribute retrieval, we propose our work on result aggregation [98], one of the novel challenges in relational aggregated search. We propose a weight-based framework to assemble relational aggregated search results. In particular, we focus on the construction of tabular

results for class queries (e.g. French wines, Macintosh laptops, Nokia mobile phones, ...). The result will have instances in the first column and their attributes (names and values) in rest of the columns. The weights are given with respect to the importance/relevance of instances and attributes.

- In addition, we propose our investigation for large-scale extraction of lists of instances of the same class [91, 92], which are proven useful for relational aggregated search. Typically, existing techniques identify them passing by the class acquisition, but this process might miss many of these lists. We propose an alternative approach that relies on HTML lists where we avoid class acquisition.
- Finally, we show 4 relational aggregated search applications (prototypes) built through our approaches. Three of the prototypes are dedicated to one type of query each respectively attribute queries (e.g. president of France, major of New York, ...), instance queries (e.g. Eiffel Tower, France, New York, ...) and class queries (e.g. countries, American cities, ...). The forth prototype can answer all three query types while it can trigger a dedicated solution for each query type. Moreover, within these prototypes we also introduce attribute value retrieval, passage retrieval and image retrieval. This work is interesting for two reasons: we show the components of relational aggregated search from an application perspective as well as we show encouraging results for relational aggregated search.

### 1.2.2 Cross-vertical aggregated search

Cross-vertical aggregated search has already a consecrated place within aggregated search to the extent that it is often used as a synonym of aggregated search [99, 14, 12]. However, its differences with federated search are not clear. Our contribution in this direction concerns mostly the interest behind this new paradigm and its evaluation.

- We propose a study [94] that investigates on the interest (advantages, new issues) of cross-vertical aggregated search. Interest in this direction has been proved through analysis on query logs [14, 104, 154] or user studies with broad tasks [166]. These studies rely on different configurations and they only prove partially the interest behind cross-vertical aggregated search. Our aim is to revisit them by exploiting two definitions of relevance (by intent and by content) and two types of queries (short text and fixed need). Our study targets the notion of source relevance and the reasons why multiple sources can be relevant at once: ambiguity, complementary results, different levels of relevance, ....

- Our research targets evaluation issues, too. In fact, interest and evaluation are strongly related. We cannot compare different approaches without identifying clearly their presumed advantages. Concretely, we analyze differences among 4 different evaluation setups. Our study identifies weaknesses and strengths of each approach for relevance assessment.

### 1.3 Thesis outline

This work is divided in 5 parts. The first part (this part) contains the introduction of this work. The second part is about background knowledge and state of the art. The third and fourth parts present our contribution respectively for relational aggregated search and cross-vertical aggregated search. The last part is about conclusions and future work. In this section, we will introduce the contents of each part.

- In part 2, we present background knowledge and state of the art. It is composed of 4 chapters. Chapter 2 introduces the broad paradigm of Information Retrieval. We describe a broad description of IR systems with the corresponding components. Then, we distinguish two broad classes of approaches namely boolean retrieval and ranked retrieval. We argue on the limits of these paradigms before introducing aggregated retrieval as an alternative and sound paradigm.

The state of the art of aggregated search in its broad conception is provided in chapter 3. We initially introduce a generic framework which allows us to decompose the aggregated search problem and analyze its approaches. Then, we list existing approaches using different perspectives. Within the different research directions, we choose relational aggregated search and cross-vertical aggregated search as the most novel and most promising in research.

In chapter 4, we present relational aggregated search which is described as a novel and promising instance of aggregated search. Here, we list the existing related work in the cross-road of many areas such as Information Retrieval and Information Extraction. In our discourse we try to highlight the most important research issues.

Chapter 5 is about another important research direction, namely cross-vertical aggregated search. We define it as a distinct of aggregated which can be classed as multi-source aggregated search. Here as well, we list the associated issues and related work.

- In part 3, we focus on our contribution for relational aggregated search. This involves attribute retrieval, result aggregation, an extraction technique for lists of instances and 4 prototypes. This work is split in 4 chapters.

In chapter 6, we focus on attribute retrieval, which is meant to be crucial for relational aggregated search. For this problem, we include the description of the approach, experimental setup and results.

In chapter 7, we consider attribute and instance retrieval as resolved problems and we focus on result aggregation. We propose a weight-based framework for this purpose which we experiment. We end this chapter with results and conclusions.

A different and complementary work is presented in chapter 8. Here, we investigate on lists of class instances already shown useful for relational aggregated search. We propose a technique for extracting them at large scale without having to identify their class. This is accompanied as well with experiments and results.

We end this part with prototypes that we could build using our research. They are described in chapter 9. We propose 4 different prototypes. We start with common issues and our solutions for each issue. Then, we describe each prototype and its components.

- In part 4, we present our research on cross-vertical aggregated search. Here, we propose a study which targets interest and evaluation of cross-vertical aggregated search. We start with our motivation. Then, we describe the study that was setup consisting of 4 simulated evaluation setups. The results are then analyzed to derive useful thoughts on the interest and evaluation of cross-vertical aggregated search.
- The last part is about conclusions and future work. This is done separately for each of the two research directions (relational aggregated search and cross-vertical aggregated search).



## Part II

### Part 2: Aggregated search



## Chapter 2

# Boolean, ranked and aggregated information retrieval

*Information Retrieval (IR)* is finding material (information) within large collections of documents (usually stored on computers) [112]. An *Information Retrieval System (IRS)* is built on top of one collection of documents to facilitate information access, organization and storage [50]. The term *search engine* is also used to refer to an IRS.

Information retrieval can help for whatever large collection of documents of unstructured or partially structured data such as text, HTML documents, XML documents, but also multimedia such as images, video, etc. The collection can be a corpus of books, a corpus of news articles from one or more news agencies, all pages of a Web site. The most illustrate example of IR system is represented by Web search engines. They are used daily from most Web surfers. Names such as Google, Yahoo!, Bing are popular to almost everyone. Their users write down what they are looking for in their own words and these search engines return a list of results potentially useful (relevant).

In this chapter, we will initially define the common elements of IR systems as well as the components of the IR process. Then, we will define boolean retrieval and ranked retrieval. Critics on these paradigms will be used to support the definition of aggregated information retrieval.

### 2.1 Information Retrieval Process

The user of an IRS has a need for information that can be potentially found in a collection of documents  $C$ . An information retrieval system (search engine) is built on top of this collection to help the user satisfy his/her needs for information. We can distinguish here three important components of the IR process:

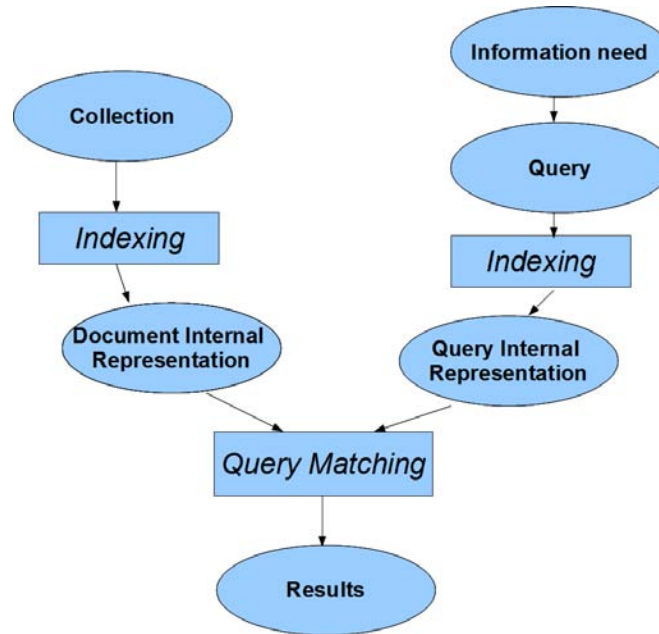


Figure 2.1: The IR process simplified schema

- **Indexing:** The IRS indexes the collection to enable fast and efficient information access
- **Querying:** The user expresses his/her information need in the form expected from the IRS, usually a free-text query
- **Query matching:** Every query is matched with the documents in the index

This process can be more complex than described, but we describe a simplified view, which is also schematized in figure 2.1 as a U-process [27]. The indexation process is not executed every time a query is issued. It is generally executed once initially and every time the collection has important updates. We will describe in the following sections (2.1.1-2.1.3) each of these components. This will allow us later to introduce the two broad information retrieval paradigms namely boolean and ranked retrieval. The latter will be described in section 2.2

### 2.1.1 Indexing

We cannot compare a query with every document in a collection as it would be far too expensive in terms of computation time. This is easy to realize if we estimate the answering time of Web search engines which are built over several billions of documents. The users of these engines would have to wait

for days or months for results. Search engines demand for multiple optimizations to guarantee that the answer will reach their users in a reasonable time with a reasonable quality.

An essential component of an IRS is the construction of an index of the collection also known as indexing. The index is a large data structure which contains data about the collection that enables fast and efficient querying. Most indexes are built from the text within documents even when the target of the IRS is images, videos, etc. In this section, we describe common indexing of text, although for specific applications and data formats, indices contain specific features such as image dimensions, video length, etc.

The most common form of index is the *inverted index*. The inverted index is composed of a *vocabulary* and *posting lists*. The vocabulary contains all terms extracted from documents retained as useful for information retrieval purposes. Every term in the vocabulary is associated with a posting list which tells in which documents the term appears. In other terms, instead of having for every document a pointer towards the terms that appear within, we keep an index of all terms pointing to the documents they appear in.

Documents need to be processed to extract only the targeted content (text, images, videos). Then terms are extracted, processed and weighted to build the index. The following steps are common in indexing text. They are applied to documents, but also to queries.

- Extracting a normalized sequence of characters
- Tokenization
- Stop-word removal
- Normalization
- Lemmatization and stemming
- Term weighting

We describe next briefly each of these steps. Some of these steps can be language-dependent. To illustrate, we will suppose English to be the language of the indexed documents.

**Extracting normalized sequence of characters:** Documents need to be processed to extract text as a sequence of characters. Concretely, documents can have different formats such as pdf, Microsoft Word format, HTML. Furthermore, the text can be in different encodings such as UTF8 or ASCII. The goal is to remove all format specific structure and keep just a linear textual sequence should in one predefined encoding.

**Tokenization:** After transforming the document in a sequence of characters, *tokenization* is applied. This corresponds to the task of chopping this sequence into pieces (usually terms), also called *tokens*. Typically, tokenization is also associated with punctuation removal. The final result of this step is a sequence of terms (tokens).

**Stop-word removal:** The presence of some terms can be of no statistical importance for information retrieval. This is the case for very frequent terms such as “the”, “an”, “a”, “of”. These terms are called stop words and they are removed in most IRS-s. The removal of stop words has some advantages and disadvantages. On one hand, it reduces significantly the index size helping answer queries more quickly. On the other hand, for some queries the presence of stop words can be of vital importance. Let’s consider the query “The Who” which refers to a famous rock band. In this case, all the query terms can correspond to stop words. This might explain the fact that some of the major Web search engines do not remove all stop words.

**Normalization:** Sometimes, we might want different tokens to match each other. This is the case for words such as N.A.T.O and NATO, or US, USA, U.S. and U.S.A. We call *token normalization* the task of grouping tokens that should match with each other in *equivalence classes*. For instance, we can map both the tokens “semi-conductor” and “semiconductor” in one of the two terms.

There are different ways to normalize. Above we mentioned some examples where it is enough to remove punctuation. Another way to normalize is to reduce all letters in lower case also known as *case-folding*. We have to be careful with case folding. If we meet Jaguar with the first letter in upper case, it is likely to be the car make, while if we meet “jaguar” in lower case it is more likely to refer to jaguar the animal. Classes of equivalence can also include synonyms.

**Lemmatization and stemming:** The same word can be meet in different forms. For instance, “computing”, “computes”, “computed” and “computation” are all different syntactic forms of “compute”. It makes sense to group words of similar meaning together, especially when they are just grammatical transformations of each other. *Stemming* and *lemmatization* are two well-known techniques that tackle this issue. Stemming is just some heuristics which work well most of the times. It mainly consists of removing some word endings (postfixes). On the other hand, lemmatization is more precise, but demand more hard-coded rules. It corresponds to the removal of inflectional endings to return the dictionary base of a word also known as lemma. One of the most used techniques is the Porter Stemming algorithm which is empirically shown to be very effective [142].

**Term weighting:** For every term in its normalized, stemmed and/or lemmatized form, indexing produces some statistics which are vital for future scoring in the query match process. We list below some important features/weights that are commonly stored in an index:

- **term frequency (tf):** The term frequency corresponds to the number of times a term appears in a document.
- **document frequency (idf):** The document frequency corresponds to the number of documents a term appears within. Usually, it is the *inverse document frequency* that is used, because a term that appears in many documents is unlikely to be discriminative.
- **term location:** It is also common to store the location of a term in a document. This can be used to detect if two terms appear consecutively in a document.

We presented above just a short summary of the steps involved in indexing. The index is very important in the query matching process. The document-query match is scored based on the statistical data stored in the index.

### 2.1.2 Querying: Information need and queries

The concept of information need is at least as old as Information Retrieval [152]. We can find early definitions in the work “The process of asking question” [171] from Robert Taylor. The author studies the way individuals obtain answers from information systems. One of the definitions he gives of the information need is of an *unconscious or conscious need for information not existing in remembered experience*.

In other terms, the information need exists in the presence of missing knowledge. Individuals address IR systems when they suspect they might find the needed information through them. Typically, they issue free-text queries i.e. users write down their needs and they give them as an input to the IRS. So far, the IR process is more difficult than that. The IRS user should be aware that the quality of the retrieved results depends on the query he issues. Most search engines deal better with short queries which contain well-selected terms.

The input to a search engine does not have to be a textual query. For instance, it can be an image and the returned result can be other similar images. Nowadays, textual queries remain by far the most used. Most information needs are expressed as free-text queries.

### 2.1.3 Query matching

In all IR systems, documents and queries are indexed and represented internally as a set of features. Most of the features are computed from the index.

Typically, each term in a document can generate a feature, usually a real number indicating the weight of the term within the document. The query is also transformed into a set of features. These representations are also known as *internal representations (IREP)* of the query and the document.

Because queries are usually quite short, it is also common to extend the query with other related terms. This process is called *query expansion*. The final feature representations of the query and the document are used for the *query matching process*.

The result of the query matching process is pairs of documents and scores where the score indicates the relative or absolute degree of presumed relevance for the document with respect to the user's information need. In boolean IR models [112], this score can be 0 or 1. The 0 score marks the document as irrelevant and the 1 score marks the document as relevant. This approach is obsolete in most of the IR applications today, because it scores all relevant documents equally and it can often return 0 results. Current IR models assign real value scores to documents. This allows ranking documents with each other.

We will now present each of these two models accompanied with some critics. We will then introduce aggregated retrieval.

## 2.2 Boolean and ranked retrieval

The different IR process can be instantiated in different ways. Within different approaches, we distinguish two broad paradigms that have been the goal of research for many years namely boolean and ranked retrieval. The main difference between these approaches is that boolean retrieval returns a set of (unranked) documents and ranked retrieval returns an ordered list of search results. Boolean represents the dominant model in the early times of IR, while ranked retrieval has taken the lead in current times. We describe below each of them.

### 2.2.1 Boolean retrieval

Boolean information retrieval dates back to the origins of IR [112]. In this model, queries are expressed through boolean logic. For instance, the query "New AND York" tells that we want both the terms "New" and "York" to appear in the retrieved documents. The query "(London OR Paris) AND NOT Berlin" would match documents that contain the terms "London" or "Paris", and none of retrieved documents will have the term "Berlin".

In this model of IR, it does not matter how often a term appears in a document, we are only interested in its presence within the document. A document that contain the term "London" 20 times scores the same with a document that contains the term "London" once. This approach has the



advantage that it shows only documents that will match the query precisely, but it has many other disadvantages. We will list below some of them.

First, sometimes it is difficult to know the exact words that will have to appear in the documents as well as it difficult to guess the best logic operators to connect these terms. In fact, most users of the boolean model were expert users. The large audience does not necessarily master boolean logic.

Second, the returned documents will have no order. If the list of returned documents is long, one might prefer to access the documents which are more likely to be relevant first. The lack of order affects both true negatives and false negatives. In other terms, let's take the query "Paris and London and Berlin" where we want to compare these cities. The results which mention these terms several times will score the same with the results which contain each of these terms once. On the other hand, a document which contains both "Paris" and "London", but does not contain "Berlin" will score 0, the same as other documents which do not contain any of the query terms.

Nowadays, pure boolean IR has turned to be obsolete for most IR applications. Existing techniques had to integrate some sort of ranking.

### 2.2.2 Ranked retrieval

Ranked retrieval [112] allows users to issue free text queries i.e. they type one or more words without any logic operators or other complex operators. The main difference with boolean IR is that the output is a ranked list rather than a set of documents. Here, we can always list results even when some of the query terms are not present in the document. The ordering of the results aims to keep relevant results on top. This corresponds to the ranking principle. Robertson states that the optimal IR system should order results by their probability to be relevant [146].

Typically, results are ranked by scoring functions which combine different features generated from the query and the documents. Instead of the binary presence of a term within a document, ranked retrieval models combine other weights. Within them, two are the more popular namely term frequency *tf* and the inverse document frequency *idf*. Features are also specific to the IR model being used where we can mention vector space models [151], probabilistic models [147, 30], language models [140, 31], fuzzy models [26].

The next section presents some critics on traditional ranked retrieval.

### 2.2.3 Limits of ranked retrieval

The list of documents of the same format is not necessarily the best approach for Information Retrieval. A survey on traditional Web search [22] shows that users find relevant search results in the first page of results in 39.9% of the cases and that only 21.2% of the interviewed find the results well

organized. Below, we list some limits on the ranked retrieval paradigm from different perspectives.

**Data sparseness:** The relevant information can be scattered in different documents [123]. The ranked list for these cases is inadequate, because the user has to scan within different documents to satisfy his information need. This can be a time-consuming and burdensome process.

**Lack of focus:** Ranked retrieval approaches provide a ranked list of uniformly presented results. Typically, each result is a snippet composed of the result title, a link to the document and a summary of the linked document. But it is known that the beginning of a document is not necessarily the best entry point [145]. For queries when the answer is just a part of document, it might be better to return this part of document right away. The uniform snippets do not have enough flexibility for focused retrieval.

**Lack of diversity:** For some queries, search results should be diverse [47] in both terms of content and presentation. The traditional ranked retrieval approach would provide a uniform presentation on all results. The queries “images of Niagara Falls”, “videos of Niagara Falls” and “Niagara Falls” would all be returned Web page snippets from traditional Web search. Ideally, the first two queries should be returned respectively images and videos right away, while the third query can be answered with diverse results (images, videos, Web pages, ...). Ranked retrieval approach should account for diversity in both terms of content and presentation. In fact, diversification of search results has an increasing interest in IR research [45, 8].

**Ambiguity:** Many queries can be ambiguous in terms of information need. The reference example is *Jaguar* which can refer to a car, an animal, an operating system and so on. Ideally, we should return one answer per query interpretation [161]. This can be multiple ranked lists or linked sets of results.

## 2.3 Conclusion: Towards aggregated retrieval

In this chapter we presented the basic architecture of traditional IR systems with some details on their main components. Within the retrieval approaches we distinguish two main classes namely boolean retrieval and ranked retrieval. We listed on both of these paradigms many disadvantages which proves interest towards new information retrieval paradigms. Within these limits we highlighted data sparseness, lack of focus, lack of diversity and ambiguity. This is sound motivation that tells for the need of new information retrieval paradigms. Aggregated retrieval represents one of the

promising alternatives, because it can incorporate more focus, diversity as well as flexible result assembly. We will describe “how” and “why” in the next chapters.



## Chapter 3

# Aggregated search

### 3.1 Introduction

Data sparseness, lack of focus and lack of diversity are just some of the limitations of ranked retrieval. To whatever query, the user is returned a list of documents. However, the user often needs just a small part of a document. Even worse, his/her information need can be sparse in many documents. In this context, it is reasonable to investigate for new information retrieval paradigms. Among these, we distinguish a new research domain, namely *aggregated retrieval* or *aggregated search*. This new paradigm concerns different ways of putting together search results. In other terms, aggregated search considers that there are ways to put results together other than the ranked list. If we see ranked retrieval as an extension of boolean retrieval and if we consider ranked retrieval as traditional information retrieval, then aggregated search is an extension of traditional IR, and we might classify IR models as boolean retrieval models, ranked retrieval models and aggregated retrieval models.

Major Web search engines already provide some form of aggregated search. They do not return anymore only lists of Web pages. They also include results of different type such as images, news, videos, definitions. Let's illustrate with some examples (some of them were already introduced earlier). The query "define brontosaurus" issued to Google<sup>1</sup> is indeed answered with a definition which is followed by a list of results. For the query "brontosaurus images", we will find around 16 images followed by a list of Web pages. On the other hand, the query "brontosaurus" alone issued to Google<sup>2</sup> will be answered with a Wikipedia article, 4 images and other Web pages. We can see that Web search provides more focused search results and that sometimes different results can be useful at the same time (e.g. images, definitions, Wikipedia articles, etc.). To do so, major search

---

<sup>1</sup><http://www.google.com> accessed in August 2011

<sup>2</sup><http://www.google.com> accessed in August 2011

engines combine results from different search engines, Web search, video search, image search, etc. From this perspective, it can be seen as a special case of federated search [38].

Although major Web search have introduced more focus and diversity within search results, there is no explicit assembly of results. The query “French red wines” is unlikely looking for a list of documents and it is difficult to find all needed information in one page. Adding images, videos or news articles will not help. Alternatively, one can put together a list of instances of French wines with their properties (attributes) such as color, region, description, price, etc. The query “all about Nokia e72” also demands for information from many documents. This can be a description, a list of features, images, videos, but also reviews, ratings, news articles, etc. Here assembly can be explicit because we are generating a new document on “Nokia e72”.

The above examples are just meant to illustrate. Aggregated search can be met in many other forms and applications. Depending on the application, the content to be retrieved and be put together can vary in terms of granularity (document, passage, sentence, phrase) and type (text, video, image).

The first definition of aggregated search (AS) is met in a dedicated workshop in SIGIR 2008 [123]: *Aggregated search is the task of searching and assembling information from a variety of sources, placing it into a single interface*. From this perspective, an aggregated search system is built on top of one or multiple search engines (sources).

We can decompose an aggregated search system into three main components namely *query dispatching* (QD), *nuggets retrieval* (ND) and *result aggregation* (RA). We will introduce them briefly here and then we will detail later in a general framework.

**Query dispatching:** We use this term to refer to the process that precedes retrieval (query matching). It includes the interpretation of the query and the selection of the sources to be used. *In aggregated search, we use the term source to refer to a search engine that relies on at least one collection and one search algorithm*. The notion of source is important here, because aggregated search systems can be classified as *mono-source* and *multi-source*. When there are multiple sources, it is up to query dispatching to select the sources to be used.

**Nugget retrieval:** As we mentioned earlier, we can retrieve content of different granularity (sentence, passage, document, ...) and different type (text, image, video, ...). To remain general, we introduce the *information nugget* concept which enable us to generalize well across different instances of aggregated search<sup>3</sup>. *The term information nugget will refer to whatever*

---

<sup>3</sup>The term “*information nugget*” has been used frequently in research to denote semantic pieces of information [45, 62, 143, 11] and in particular in question answering [88, 180],

*amount of data which can satisfy some information need.* This can be a document or a part of document. Though, an information nugget can be a single word such as “Paris” when answering “capital of France”, but it can also be a sentence, some sentences, a paragraph, a document, a link, an image, a video, etc. The nugget retrieval corresponds to the identification of the potentially useful information, but it does not involve their assembly.

**Result aggregation:** Given a set of potentially relevant information nuggets, result aggregation corresponds to the assembly of these nuggets into a final answer. Depending on the cases, this can involve summarization, ranking, clustering, . . . . We will provide a generalized view of result aggregation actions in section 3.3.3.

In the next sections, we will list the broad issues of aggregated search. Then, we will detail the general framework for aggregated search composed of nugget retrieval and result aggregation. Then, in section 3.4 we will provide an overview of aggregated search approaches from different perspectives.

## 3.2 Issues

If we keep to the broad problem of aggregated search, we can list some of the broad and important issues in this domain, although they might sound a little abstract initially.

- **Identify the type of answer the query demands for:** In fact, different queries can be answered differently. Some queries can be answered with a single information nugget, some others with multiple information nuggets. Queries such as “capital of France”, “BBC homepage”, “height of Everest”, “definition of Brontosaurus” can be answered with a single information nugget, while queries such as “French wines by region”, “ratings of Nokia e72”, “chinese restaurants in New York”, and “all about Nokia e72” demand for multiple nuggets. We can observe that the way information nuggets should be assembled depends on the query. It becomes though important in aggregated search to integrate more flexible query answering.
- **Identify the components (information nuggets) of the final answer:** In aggregated search, we can retrieve information nuggets of different granularity and different type. This enables more focus in the final answer. It is not trivial though to identify the information nuggets that should be used to compose the final answer. When should we use part of a document instead of an entire document? When should we use multimedia content (images, videos, . . .)? When should we use

---

although without any common agreement on its meaning. Here, we give a general definition which suits also the context of aggregated search.

specialized search engines (image search, video search, news search, ...)? This is one of the most difficult question in this domain.

- **Assemble together the different information nuggets in a coherent aggregated document:** Aggregated search can involve all possible ways to assemble search results. This can be a summary, two images and a definition, a relational table, etc. One of the targets of aggregated search is to choose the best result aggregation given the available search results. Whatever the final result might look like, it has to be readable and coherent. The main question is how to assemble and how to evaluate aggregated results with respect to the query knowing that it is impossible to build a priori all potentially useful combinations of results, i.e. this problem is topic-dependent.

### 3.3 A generic framework for aggregated search

As we discussed before, aggregated search can be met in different forms. We have defined a general framework and conceptual schema that eases the introduction of the components of aggregated search for whatever approach.

Figure 3.1 presents a conceptual schema for the aggregated search process. It tells that we can have more than one source. The query can be issued across multiple sources to retrieve potentially useful information nuggets, which is similar to federated search [38]. However, this is not a necessary condition. Many aggregated search systems can be built with one source. In both cases, retrieved information nuggets have to be assembled with each other to provide the final answer to the user. This can be whatever sensed organization of information (though not just the ranked list).

In the schema, we distinguish three important components for aggregated search: *query dispatching (QD)*, *nuggets retrieval* and *result aggregation (RA)*. In this general conception, the aggregated search system adds two additional steps to traditional retrieval namely query dispatching and result aggregation, while nugget retrieval plays the role of the information collector.

Let's consider the query "visit Eiffel Tower" and an aggregated search system which includes Web search, map (geographic) search, weather search, image search and news search. The aggregated search system can select as useful all these sources except of news search. It can issue the query "Eiffel Tower" to map search, Web search, image search, while it can issue the query "weather in Paris, France" to the weather search tool. All these steps are done before any query matching and they correspond to what we call *query dispatching*.

The *nugget retrieval* takes a query as an input and it returns a set of information nuggets typically associated with matching scores. Some sources return a ranked list of results while others return only precise matches. For



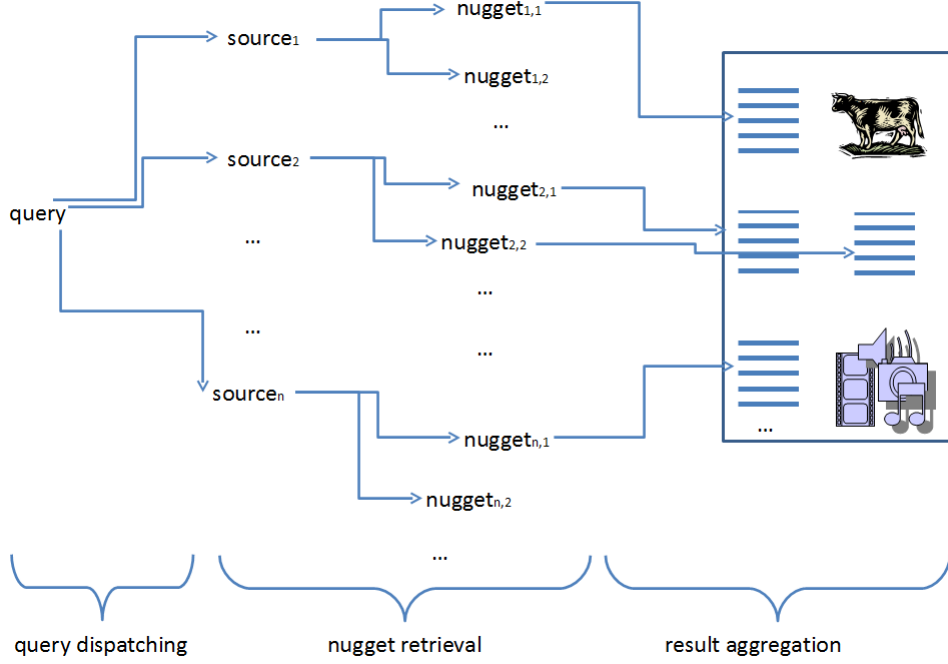


Figure 3.1: Detailed schema of the aggregated search process

instance, the query “Eiffel Tower” if issued to map search, it will likely be answered with one result. If it is issued to Web search, it will be answered with many search results. The nugget retrieval can vary a lot. Generally speaking, it corresponds to the query matching process on a given query by one or multiple sources. It is the output of nugget retrieval that will be used as input for *result aggregation*. Result aggregation has to deal with the assembly of search results.

In the next section, we will describe each of the three components.

### 3.3.1 Query dispatching

Query dispatching precedes retrieval (query matching). It corresponds to initial interpretation and treatment on the query. This involves different sub-tasks: it can select the sources to send the query to; it can trigger a specific solution; it can transform the query into one or more modified queries. We distinguish 2 sub-tasks that fall in the query dispatcher’s responsibility.

**Source selection:** Source selection is one of the most well-known problems in aggregated search. Given a set of sources, its goal is to select the sources that are likely to answer the query. We will discuss this task more in details when we will treat multi-source aggregated search.

**Query decomposition:** Some queries can be decomposed in two or

more subqueries eg. “Paris museums and monuments” can be decomposed into “Paris museums” and “Paris monuments”. If we are unable to find in one page information about both Paris museums and monuments, we can join the results of the two subqueries. We will call this kind of queries as compound queries. The same information need can sometimes be expressed with different queries which cannot be easily decomposed. eg “what to visit in Paris”. Although query decomposition is one of the query dispatching issues, it remains one of the most difficult problems in IR for some simple reasons. Machines do not know the information need behind the query and it is difficult to interpret or decompose queries expressed in human language.

### 3.3.2 Nuggets retrieval

Nugget retrieval is situated between query dispatching and result aggregation. It takes as input a query and it returns a set of potentially relevant information nuggets. Generally speaking, this process involves at least one collection of documents and at least one query matching process. This process can vary a lot depending on the type of content being retrieved and the query matching approach. We will enumerate here some of the main classes of approaches.

We can retrieve entire documents or parts of documents. This corresponds to document retrieval and focused retrieval respectively [61, 83, 175, 137, 138]. The retrieval process also depends on the multimedia type (text, image, video, ...). We can distinguish here textual retrieval and multimedia retrieval [103]. Some search engines such as Web search engines retrieve content of heterogeneous type (Web pages, images, news articles, videos, pdf files, ...). When the one-size-fits-all solution does not work well, it is common to derive vertical search solutions which are specialized on a media type, query type, task, result type, etc. We can see that there are many different ways to retrieve information nuggets.

Nugget retrieval can also involve multiple sources. This has been an intensive area of research for at least two decades. It starts with *federated search* (Distributed Information Retrieval) [38] in the context of distributed data collections (hidden Web, databases, etc). Then, it has evolved into new areas such as *meta-search* [162] and more recently into *cross-vertical aggregated search*. The latter represents the most successful class of approaches and it is applied by almost all major Web search engines. To distinguish between these approaches it is important to have a clear definition of source. The terminology in literature is a little messy as terms such as search engine, resource, collection are often used as synonyms. We recall that we prefer using the term source to refer to a search engine or a component of a search engine that uses at least one collection and one search algorithm. This enables us to classify multi-source retrieval approaches such as meta search [156, 113, 162], federated search [38, 66], data fusion [181, 191, 192, 23, 56],

mashups [67, 144] and cross-vertical aggregated search [14, 53, 123, 166, 95]. We will describe them more in details in section 3.4.

To conclude, we can say that the nugget retrieval output depends on the type of sources and the number of sources being used. This affects the focus and the diversity within retrieved information nuggets.

### 3.3.3 Result aggregation

Result aggregation starts from the moment we have a set of potential relevant information nuggets for a given query with or without relevance scores. Ranked retrieval approaches just show these results ranked by relevance, but there are many other ways to assemble results with each other.

Result aggregation can be done a posteriori (i.e at query time) or a priori (i.e. the query is not known yet) [90]. A priori result (content) aggregation would decrease the query response time, while the a posteriori approach would allow more flexibility. In general a priori result aggregation is not easy. First, it is impossible to generate all possible useful answers. Alternatively, we can generate all combination of results. Let  $n_1, n_2 \dots n_m$  all information nuggets that can be identified within a collection. There are  $2^m - 1$  ways to put together these nuggets. Even if we limit the maximum number of nuggets in an aggregated document to be  $p$  we would have  $\binom{m}{p}$  ways to assemble results. We can see that the result space can explode exponentially if we perform a priori aggregation of content. To avoid the explosion of the answer space, most of the aggregation is done a posteriori, although some content can be aggregated offline at indexing time to enable faster answering time.

Due to a survey we did on different instances of aggregated search [95], we identified several techniques that help assembling search results. Within these techniques there are 4 basic actions that concern most result aggregation approaches. They are *sort*, *group*, *merge* and *split/extract*. We will detail each of them below.

- **Sorting:** Given a set of information nuggets  $n_1, n_2 \dots n_m$  the *sorting* action produces another list of nuggets  $n_{l_1}, n_{l_2} \dots n_{l_m}$  where all elements are ranked with respect to some feature. This feature can be relevance scores, but also time, author, location, popularity scores, and so on. Sorting can also take into account diversity and novelty of search results following the principle “A result should not only be relevant, but also novel [33]”.

Although we mention that the goal of aggregated search is to go beyond the ranked list approach, ranking (sorting) by relevance or by other features remains one of the basic and most important actions that an IRS can perform.

- **Grouping:** Given a set of information nuggets  $n_1, n_2 \dots n_m$  the *grouping* action produces groups (sets) of nuggets  $G_1, G_2 \dots G_i$  where elements of the same group share some property. A group of results can be composed of results with similar content, results that have happened in the same period in time, results with a common feature, and so on.

Among grouping approaches we can mention clustering [76, 190] and classification [112] as special and illustrative cases. Clustering groups set of documents/nuggets into subsets or clusters. The goal is to produce coherent clusters while keeping dissimilar content in distinct clusters. In classification, we are given a set of pre-defined classes such as for example “sport”, “economy”, “art” known also as labels or categories. The goal of classification is to associate documents to classes. Its main difference with clustering is that each group has a pre-defined label.

We can also mention here faceted search which includes grouping. In faceted search [29], it is common to extract features/labels from the collection and results are browsed based on common features of the results such as year, author, etc.

- **Merging:** We refer to *merging* as an action that takes a set of information nuggets  $n_1, n_2 \dots n_m$  and produces a new cohesive aggregation unit (see figure 3.2). From this definition, merging is different from grouping in that it produces one final grouped unit and not multiple groups.

For instance, multi-document summarization [51, 115, 64, 120] is a form of merging, which produces one new entity, the summary. Instead of accessing multiple documents, a summary provides users a quick insight of the available content. This is different from grouping, which would not provide one final group but different groups. Typically the summarized documents are sensed to relate to some topic or to each other. That’s why some clustering might precede summarization [158].

More instances of this action can be met in section 3.4. Generally speaking, merging can be seen as an aggregation step that aims providing more relevant and related information at once.

- **Splitting/extracting:** Given some content, we can do the opposite action of merging. We can decompose the content in other smaller nuggets (see figure 3.3). The result of this decomposition can be one smaller nugget or a set of smaller nuggets. Given some content  $n$ , the result of extracting/splitting is a set of information nuggets  $n_1, n_2 \dots n_m$  with  $m \geq 1$ . Typically splitting produces total partition over the initial content such that  $n_1 \cup n_2 \cup \dots \cup n_m = n$ . On the other hand, extracting

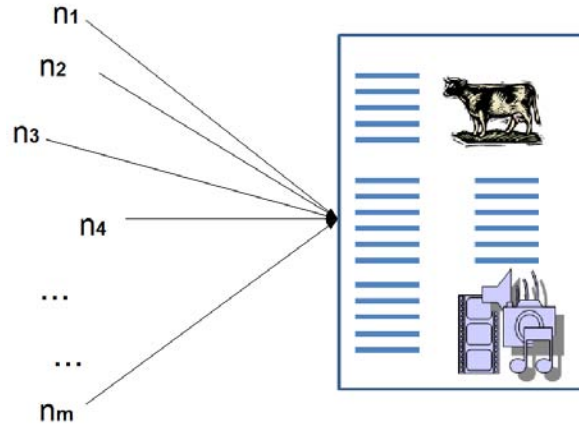


Figure 3.2: A simple schema for merging

is more about identifying one or more semantically sound information nuggets within the initial content.

Splitting/extracting is a fundamental action for result aggregation. Suppose we want to put together all restaurant names in the city of New York. We should be able to extract these names within documents and then assemble them. Or let's consider the query "Nokia e72" and let's suppose that all relevant content resides in 2 Web pages  $d_1$  and  $d_2$ . Page  $d_1$  contains the mobile phone's specifications and other irrelevant content. Page  $d_2$  contains images, videos and also some irrelevant content. To build an ideal answer, we should be able to decompose the Web pages  $d_1$  and  $d_2$  into smaller parts and then merge the relevant information nuggets.

In the next section, we list different instances of aggregated search seen from different perspectives. All of them can fit in the general framework, although each approach has its peculiarities.

### 3.4 Aggregated search from different perspectives

In this section, we will illustrate aggregated search instances from different perspectives. Each of the perspectives corresponds to one research area or the conjunction of multiple areas. This includes focus-oriented approaches which concern question answering, natural language generation, relational aggregated search as well as approaches where we assume the presence of multiple sources such as federated search, meta-search, data fusion, mash-ups, cross-vertical aggregated search. We end with some domain-specific applications.

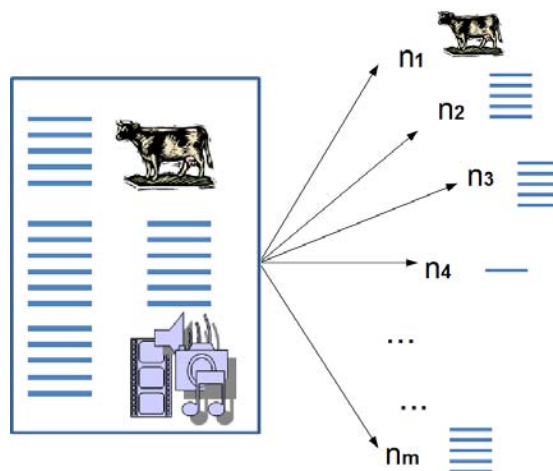


Figure 3.3: A simple schema for splitting

### 3.4.1 Question Answering

Question answering (QA) addresses question-like queries and it aims focused answers for them [88]. This involves detecting the type of query (question) and then the type of answer to return. We can list some of the common query types: factoid questions (who, when, where, ...), boolean (yes/no) questions (i.e. Is Airbus based in Toulouse?), definition questions (what is . . .), list questions (Which are the names of the last 10 US presidents?).

For some queries, the answer can reside in one document. For instance the query “Is Airbus based in Toulouse?” can be answered by some document that contains the sentence “Airbus headquarters are in Toulouse.”. In this case, the question answering system should say “yes” and preferably highlight the supporting question to the answer. In general, there can be many candidate answers and many candidate supporting sentences. For some queries, it is also necessary to identify multiple components for the answer. This is the case for list questions or more complex questions such as “why” and “how” questions.

In [121], Moriceau et al. combine syntactic information with QA techniques. Their general approach is simple. They extract candidate answers, which they associate with supporting passages. Each answer is then scored and ranked. This work is particularly interesting because it discusses links of QA with result aggregation. In particular, they highlight the need of result aggregation to answer list questions, questions that can have multiple answers, and complex queries (“how” and “why” queries). We state some of these considerations. For “how” and “why” queries, the information is likely to be sparse in multiple documents. For some other queries, it is necessary to decompose the answering process. For instance, “What minister com-

mitted suicide” can be decomposed into “who did commit suicide” and “is he/she a minister”. For such queries, information can be sparse in multiple documents, too.

A different approach is presented in [189] by Wu et al. Instead of returning a list of documents to answer questions and instead of very focused answers, they propose different intermediary approaches which can assist the answering process. This corresponds to 5 different result aggregations: *(i)* an abstract and a document; *(ii)* sentence fragments with keywords highlighted and a document; *(iii)* an abstract with one document and other related documents; *(iv)* sentence fragments and multiple documents; *(v)* a set of paragraphs.

In QA, we can identify all broad components of aggregated search: query dispatching, nugget retrieval and result aggregation. The first corresponds to the detection of the question type and the question target. The second correspond to sentence, passage or document retrieval. Third, it is needed to extract semantic information from documents and assemble them in one answer.

### 3.4.2 Natural Language Generation

Given an information need (usually a question), the goal of NLG is to generate an answer with the right information in an appropriate linguistic form [130]. This can demand merging content from different documents. So far, natural language generation addresses a limited range of information needs and research focuses on application specific needs.

NLG perspective addresses issues such as the comprehensibility of the answer [68], coherence, structure of the answer. McKeown [116] recognizes the need of a prototypical structure for certain queries. Returned facts can be organized based on some relationships. For instance, they can be ordered chronologically, they can have a cause-effect relation, background information is shown first and so on [134].

Paris et al. [132, 133, 131] show benefits and application of natural language generation approaches across different domains. In [132], they use discourse planning to generate automatically surveillance reports tailored to various contexts and tasks. The document generation is considered as a goal which can be decomposed in sub-goals represented through a discourse tree. Nevertheless, the information need in this application is not explicit (query), but implicit. In [133], authors use a similar approach for a traveling application. In this work, they incorporate query-based search. The user can indicate his target destination, but also some budget constraints. The third application is called Scifly and it produces brochures on demand. The users query is the name of an organisation and the result is a generated brochure containing several text passages. The effectiveness of this approach is evaluated through user studies.

Instead of summaries, Szlávik et al [169] aggregate content from different documents which they show preceded by a table of contents. This approach facilitates navigation, but it does not guarantee any coherence among the assembled content.

Sauper et al. [155] propose an approach to automatically generate Wikipedia-like documents. They study the document structure for specific classes of information such as diseases and actors from Wikipedia<sup>4</sup>. The learned document structure is then used to automatically build inexistent documents for other instances of the class. For instance, for diseases it is common to have sections on causes, treatment, symptoms. For a disease which is not present in Wikipedia, a document with the same structure is generated extracting information from the Web.

To summarize, natural language generation approaches also fall in the general aggregated search framework. We can observe that queries can dispatch different solutions. It involves various nugget retrieval and result aggregation.

### 3.4.3 Relational aggregated search

In this class of approaches, we place a generalization of relational search [35] and entity-oriented search [19]. We consider it as a search paradigm that relies on relations between different information nuggets. In addition to nugget retrieval, relational aggregated search retrieves relations. The latter can be precious for result aggregation.

To better understand this new paradigm, we will first present entity-oriented search and relational search. The first one is more about retrieving entities, while the second is more about retrieving their relations. The combination of both enables what we call relational aggregated search.

- **Entity-Oriented search**

Named entities are common concepts which belong to categories such as locations, person names, organisations, .... They are also called class instances [9, 93]. They are particularly common in text and queries. In a recent work, Kato et al. [87] found that about 71% of the Web search queries contain named entities. Another recent study [25] on query logs found that about 73% - 87% of the queries contain named entities and that about 18% - 39% of the queries are named entities. Given the importance of named entities and their frequent occurrence in queries, there is an increasing interest in retrieving them as well as retrieving content for them.

Instead of a list of documents, in entity-oriented search the result is a list of entities [19]. This is useful when we cannot name some entity

---

<sup>4</sup><http://en.wikipedia.org/>



(e.g. actor playing in *Pretty Woman*) or when we want to find related entities (e.g. entities similar to Nokia e72).

From a broader perspective we can consider that information is aggregated around entities. When we query about the entity, we can then return a lot of concerning information about it. In literature, there exist plenty approaches that take an entity as a query and return related content such as the Wikipedia homepage of the entity [19, 20], images [170], social network profile of a person [195], etc.

In [24], authors define the notion of composite item to correspond to the conjunction of an entity and related compatible entities. For example, a user shopping for an iPhone can be presented as a composite item containing the iPhone and a list of gadgets that match the iPhone, all within the user’s budget. The approach is interesting but authors do not focus much on the retrieval process rather than on aggregation with given constraints.

Another name for entity-oriented search is object-level search [126, 125]. In the latter, the main target is to extract and assemble information around an entity. The result of this aggregation is referred to as object. This enables returning pre-built objects instead of documents.

- **Relational search:** In Information Extraction (IE), it is common to extract and relate content from documents. Existing approaches can extract named entities such as person names, locations, organisations, etc., but also their relations such as “John” *works for* “Motorola”. Information retrieval based on these extracts is also known as *relational search*.

In [35], authors identify different types of queries that can be answered with relational search. To illustrate we can give some examples such as “French wines”, “capital of France”, “features of iPhone” [93]. The first query can be answered with a list of instances (named entities). The second query can be answered with an attribute value, while the third can be answered by many attributes (name and value).

Relational search is enabled by information extraction techniques [9] and mining within semi-structured data [36]. Existing techniques can discover many information extracts and their relations. Nevertheless, their use for information retrieval remains limited.

Because retrieving information nuggets and their relations is possible and because this can enable flexible result aggregation, we consider relational aggregated search as one of the most promising research directions of aggregated search. We will study in detail relational aggregated search in chapter 4.

### 3.4.4 Federated search

Most of the work in IR with multi-sources is classified as *federated search* [18, 6, 79] also known as *distributed information retrieval* [38] or *text database discovery* [66].

In federated search, instead of having one central collection indexed by one search engine, there are many distributed collections each indexed by a search engine. In figure 3.4, we illustrate the difference between a federated search system and a simple search engine. In federated search as in all multi-source aggregated search, query dispatching, nugget retrieval and result aggregation are always present.

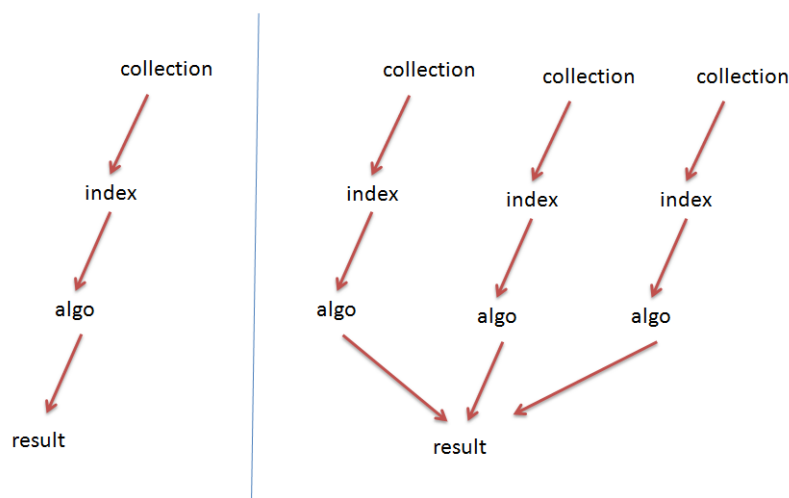


Figure 3.4: Schema for a simple search engine and a federated search engine

For instance, in the FedLemur project [18], we are presented a federal search engine that is built on top of statistical data issued from 100 US federal agencies. Instead of building a centralized collection which can quickly get outdated, authors propose building local search engines in many distributed nodes (one per agency). The federated search system has to select at query time the relevant sources, query them and assemble in one list the results.

We can say that the collection of data  $C$  is sparse in many sub-collections  $C_1, C_2, \dots, C_l$  such that  $C_1 \cup C_2 \cup \dots \cup C_l = C$ . For every collection  $C_i$  there is a source  $s_i$  which is wrapped through some API. At query time, the federated search system has to select the sources which are likely to be useful for the query. To do so, local representations of significantly reduced size of each collection are used. The obtained results from different sources are then to be assembled with each other. Typically, the final answer is a ranked list.

Although there is a significant amount of research in federated search,

it has met little commercial success in its initial days. Federated search approaches with multiple distributed collections have not been shown to produce very significant improvement with respect to centralized approaches. Nevertheless, they are commonly used when the sources are too heterogeneous or when some collection is hidden behind some API.

### 3.4.5 Meta-search

Meta-search can be seen as an instantiation of federated search in the context of Web search [99]. Initial meta-search engines used to wrap different Web search engines with the goal of improving precision and recall of Web search [156]. This was reasonable at the time because the indexes of the existing Web search engines covered small fractions of the Web. Though, the chances to have relevant results in another source were high.

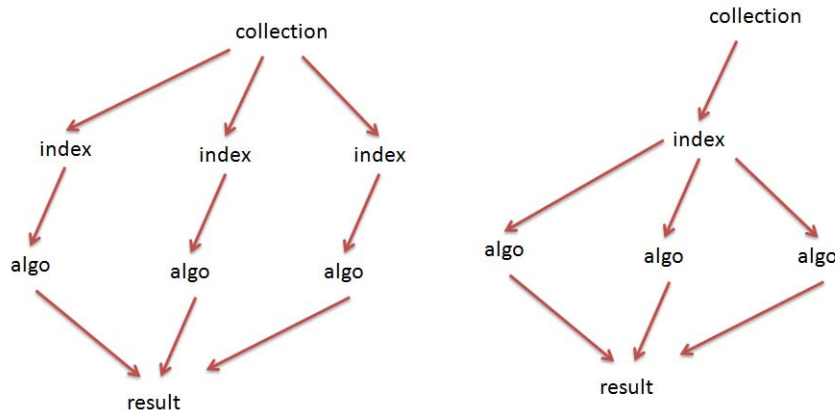


Figure 3.5: Schema for meta-search and data fusion

The sources in meta-search are typically black-box search engines which receive queries and return a ranked list of results. The returned results from different sources are then combined into one interface [156, 113, 162]. Typically results are sorted by source, but they can also be ranked with each other in one list. In contrast with most federated search work, sources can target the same task and collection. This is the case for Web search meta-search engines which are built on several existing Web search engines, each of them targeting “Web search queries”. A simple schema is shown in figure 3.5 on the left.

Meta-search engines have not known a significant success. This can be explained by the advancement in major Web search engines which have now very huge indexes and well-performing algorithms. Furthermore, there are important usage limits in Web search API-s giving meta-search few chances to become competitive.

### 3.4.6 Data fusion

The data fusion problem in IR is about finding optimal combinations of search algorithms. The idea is that common ranking functions have their weaknesses and strengths. One algorithm might work well for some query and fail for another. The goal of data fusion is to combine different ranking functions  $f_1, f_2, \dots, f_m$  in a new ranking function  $f'(f_1, f_2, \dots, f_m)$  that improves global performance of ranking. The output of data-fusion algorithms remains a ranked list. A simple schema is shown in figure 3.5 on the right.

The investigated approaches are quite rich. In [102], Lee combines different evidence through some heuristics. Its method remains one of the best performing. Wu et al [191] extends this work with the addition of weights for the ranking functions. Montague et al. treat data fusion as a voting problem [119]. The above is just some examples of various approaches for data fusion.

Data-fusion can be seen as a multi-source AS problem if we consider each ranking result as a different source of information. Data fusion is also used as synonym of meta-search [56]. The main difference in meta-search is that different sources can use different collections, while in data fusion the collection is fixed and it is the combination of ranking functions that matters.

### 3.4.7 Mash-up

Mash-ups represent an interesting approach which relates to the ways one can interact with information systems and information retrieval systems. They represent tools composed of several resources (search engines, databases, ...) assembled sequentially or paralelly (see figure 3.6). The output of one or many resources is the input of another resource. Most of the mash-up applications are domain specific and they are hard-coded [67].

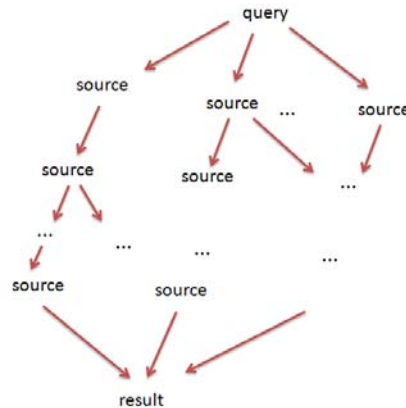


Figure 3.6: Schema for mash-up

Each resource is a blackbox which provides some API to enable querying and result re-use. These resources can be search engines, but also databases and other services. For instance, let's have the yellow pages database as well as an image search source. Given a person name  $P$ , we can generate an SQL select query to retrieve personal details for  $P$  and through the image search engine we can retrieve a list of images in some markup language such as XML. The mash-up system can use the output of both resources to generate a document similar to a visit-card.

Although mash-ups are quite promising, they need quite some integration effort and their applications are typically domain-specific. This is why mash-ups are usually oriented toward expert users [67, 144].

### 3.4.8 Cross-vertical aggregated search

Cross-vertical aggregated search [14, 53, 123, 166, 95] deals with the aggregation of search results from different vertical search engines. This is usually done in a Web search context. A vertical search engine can be image search, video search, news search, etc. Aggregated search allows users to query different verticals and Web search from the same interface. *We define as cross-vertical aggregated search the task of searching and assembling information from vertical search engines and Web search.* A simple schema is shown in figure 3.7.

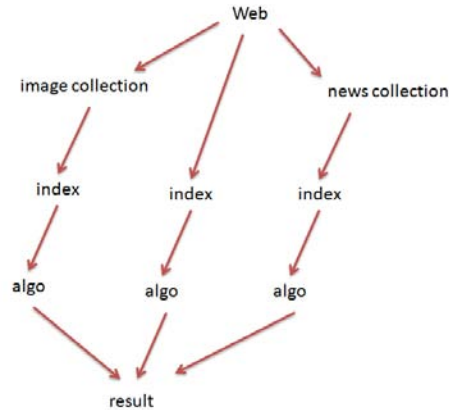


Figure 3.7: Schema for cross-vertical aggregated

Cross-vertical aggregated search provides more visibility to vertical search engines and it integrates its advantages within Web search. A vertical search result can often be better than the Web search result. Furthermore, sometimes the relevant content can be sparse in different sources. For instance “Inception movie” can be answered with news, videos, images, wikipedia article, its homepage, etc. From this perspective this research direction represents an interesting case study for query dispatching, nugget retrieval and

result aggregation. The vertical search engines can be assigned specific tasks and the final result can be more than a list of nuggets.

Cross-vertical aggregated search is met in literature as instance of both federated search[6] and meta-search [70]. Nevertheless, since the definition of aggregated search, a lot of recent work [14, 99, 166, 94, 95] is classified within this later direction.

We will concentrate on cross-vertical aggregated search in chapter 5.

### 3.4.9 Domain-specific applications

Instances of aggregated search can be found in domain-specific applications. These approaches can sometimes be too specific, but they remain interesting to study, because some of the work can be generalized to larger use.

In [85], Kaptein et al. investigate focused retrieval and result aggregation on political data. Their documents are long (50-80 pages) containing transcripts of the meetings of the Dutch Parliament. Most of the documents have similar known structure and they have meta-data such as topic, speaker, interruption, year . . . . Instead of returning entire documents, authors choose speeches as best entry points into documents. For a given query, they provide a summary of the results as a graph with 3 axes: year, political party and number of search results. Search results can be browsed through three facets: person, political party and year. Each speech is summarized in both terms of structure and content. The latter corresponds to word clouds and interruption graphs. We can observe at least 4 different aggregation formats within this work namely interruption graph (structure summary), content summarization, facets, results graph with 3 axes.

Another application is met in social science [127]. Here, authors provide search across research papers. They extract and aggregate research concepts, their relations, research methods and contextual information. The results can then be browsed by method, relation or research concept. For each research concept, the user is provided a summary of contextual information.

Strotmann et al. [163] also focus on research articles. They introduce two graph-based structures to help browsing search results. The first is a graph on the papers aggregated by author. The second is a graph of the authors with links based on co-citation analysis.

We will also illustrate domain-specific instances of aggregated search through two case studies namely *news search* and *geographic information retrieval*. We selected them for being interesting and well-known to the large audience.

- **News search:** News represents one of the primary sources of information especially when it comes to actuality. With the massive publication of news articles in many sites by different news agencies, it becomes necessary to provide search functionalities over news. News

search should adapt to the massive and frequent publication of new content.

To be able to cover many different events and points of view, search engines crawl news articles from the sites of different news providers, while news meta-search engines rely on different news search engines [107].

Results can be grouped based on topical similarity and time. Clustering news has been shown to be beneficial [149, 77]. News articles with similar topic and near publication time can represent a news story with the concerning coverage. Such an organization can help the user to focus his search within a topic and a time interval.

Related multimedia content can be juxtaposed to a news story [148]. This is the case for Google News<sup>5</sup> (see figure 3.8) and Ask News<sup>6</sup>. Rohr et al. [148] propose a timeline to show the evolution of the topic.

News aggregators represent good examples of result aggregations that go beyond the ranked list visualization. They organize search results in search stories making use of topical similarity and content freshness.

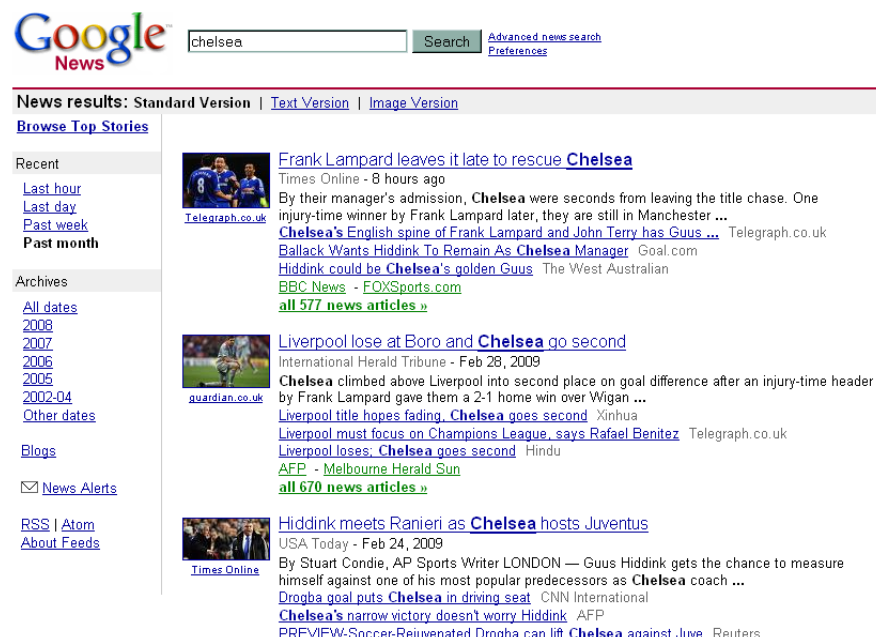


Figure 3.8: Google News results on the query “Chelsea”, accessed on April 2009

<sup>5</sup><http://news.google.com/> accessed in April 2009

<sup>6</sup><http://www.ask.com/news> accessed in April 2009

- **Geographic Information Retrieval:** Geographic location is becoming an important feature for Information Retrieval [177, 82, 153]. Information relates to geographical location as things happen in a determined geographic place. Persons and their tasks relate to their positions. This relation becomes important when searching for geographic entities as well as to personalize search based on user's location [73, 122, 32].

Geographic entities or geographic references represent physically continuous entities with static positions in geographic space [172, 82]. Examples include shops, bars, restaurants, cities, countries and landscapes. Geographic entities can be associated with geographic coordinates and shown in maps, which is particularly useful when the user is trying to locate a place or some content.

Geographic search or local search allows search for geographic entities through free text queries [177, 82]. Usually a list of candidate results is returned unless the query is specific enough to identify exactly one result. For instance, the query “Café Madrid” can identify bars in the Spanish capital, but also a bar in “Padua, Italy”. Ambiguity is an important issue, but some results are more probable than others. However some other queries implicitly demand for many results such as for “student bars in Padua”.

Typically the list of results is shown in a map juxtaposed with the corresponding ranked list. This is the case for major search engines such as Google Maps<sup>7</sup> and Yahoo Maps<sup>8</sup>. This is a form of result aggregation which allows users to localize search results in geographic space assisted by ranking which helps users identify the most useful results. Geographic proximity can be used to group and organize search results [32, 114].

If the query is good enough to identify one single geographic entity or the user clicks on one of the proposed results, a geographic search engine can support the user with more information about the entity. For instance, for a hotel it is possible to provide a phone number, reviews, images and so on.

Geographic entities can be associated with other types of content. They can be associated to images [124, 89], to related named entities [178], news articles and so on. Such relations can become useful for other vertical searches or Web search.

To summarize, in Geographic IR we can find interesting result aggregation approaches. Geographic entities can be associated to other

---

<sup>7</sup><http://maps.google.com/> accessed in May 2010

<sup>8</sup><http://maps.yahoo.com/> accessed in May 2010



content such as entities, images, reviews or Web pages. This is useful to enrich and organize search results. Furthermore, geographic related results can be shown in a map, they can be grouped by location proximity, etc.

## 3.5 Conclusions

In this chapter, we presented aggregated search starting from its broad definition. We proposed a general framework which enables to decompose and analyze aggregated search through smaller sub-problems. Further, we present the rich and heterogeneous spectrum of approaches, where it is shown that aggregated search can be instantiated in different ways. Among these approaches, we argued that cross-vertical and relational aggregated search are the most promising and fertile research directions. They can both address many queries and they can be used at large scale. Because our contribution will also concern these two types of aggregated search, we will focus on them in the next two chapters.



## Chapter 4

# Relational aggregated search

### 4.1 Introduction

Traditional information retrieval does not take into account for relations. Documents are just represented as bags of words and they are matched uniquely to the user query. Consequently, retrieved search results are just a list of unrelated items sometimes difficult to explore. Different search results are not really merged with each other. It is up to the user to select the information he needs and to assemble it. But, as we mentioned earlier, information can often be decomposed into smaller information nuggets and information nuggets can be put in relation with each other. We refer to retrieval based on information nuggets and their relations as *relational aggregated search*.

To illustrate, we will use two existing Web applications namely Google Squared<sup>1</sup> and Wolfram Alpha<sup>2</sup>. In figure 4.1, it is shown the result returned by Google Squared for the query “arctic explorers”. Each line corresponds to an arctic explorer. For each of the explorers there are attributes such as date of birth, date of death, but also an image and a description. The user can specify his own attributes to search for and he/she can search for a new arctic explorer which is not in the list. To answer this query it is necessary to rely on class-instance relations (arctic explorers - Roald Amundsen) and instance-attribute relations (Roald Amundsen - date of birth). We should highlight that information comes from many different documents.

Wolfram Alpha is another interesting case study. It exploits an underlying knowledge base with encyclopaedia-like knowledge to answer different queries. These queries can be named entities such as “Russia”, “Oslo”, “crocodylus”, “Frank Sinatra”, ..., but it can also answer queries on their attributes such as “capital of Russia”, “internet code of Russia”. In figure 4.2 we see the search result for the query “Russia”. It can involve

---

<sup>1</sup>ex <http://www.google.com/squared/> no longer available online within Google labs

<sup>2</sup><http://www.wolframalpha.com/>



The screenshot shows the Google Squared interface. At the top, there is a search bar with the text 'arctic explorers' and buttons for 'Square it' and 'Add to this Square'. Below the search bar, the results are displayed in a table format. The table has columns for 'Item Name', 'Image', 'Description', 'Date Of Birth', and 'Date Of Death'. There are four items listed: Roald Amundsen, Douglas Mawson, James Clark Ross, and Henry Hudson. Each item has a small thumbnail image and a brief description. At the bottom of the table, there are buttons for 'Add items' and 'Add next 10 items'.


Item Name	Image	Description	Date Of Birth	Date Of Death
Roald Amundsen		Roald Engelbregt Gravning Amundsen (pronounced [ˈrɔd ˈɑmʊnsɛn], 16 July 1872 – c. 16 June 1928) was a Norwegian explorer of polar regions. ...	1872	June 1928
Douglas Mawson		"Douglas Mawson" Australian Dictionary of Biography. <a href="http://www.adb.online.anu.edu.au/biogs/A100444b.htm">http://www.adb.online.anu.edu.au/biogs/A100444b.htm</a> . Retrieved on 2007-10-01. ...	5 May 1882	14 October 1958
James Clark Ross		James Clark Ross, born in 1800, entered the Navy at 11 years of age. During his first years of service he was tutored and watched over by his uncle, ...	April 15, 1800	April 3, 1862
Henry Hudson		Henry Hudson (d. 1611) was an English sea explorer and navigator in the early 17th century. After several voyages on behalf of <a href="http://en.wikipedia.org">en.wikipedia.org</a>	1570	1611

Figure 4.1: Google Squared result for “arctic explorers” accessed on May 2010

different attributes, but also maps, images, etc. The disadvantage of this approach is that it cannot answer queries which fall out of the knowledge base. Google Squared which retrieves from the the Web can potentially answer more queries and can be more up to date, although the answers are often inaccurate. Nevertheless, both applications show that we can foresee another way of assembling search results based on information extracts and their relations.

In literature [35, 37, 93, 96], the term *relational search* is used to refer to Information Retrieval based on semantic information extracts. We generalize this term to *relational aggregated search* (RAS) to refer to Information Retrieval based on information nuggets and their relations.

In the next section we will situate relational aggregated search in the general framework of aggregated search and we will enumerate some issues which we will be followed with related work.

## 4.2 Framework and issues

This paradigm fits in our general framework we presented for aggregated search in chapter 3. We can find here the same components (query dispatching, nugget retrieval and result aggregation) with an additional relations layer.

**Nugget retrieval:** Every information nugget can be used for relational aggregated search. We can retrieve and relate images, videos, news articles, etc. However, there are 3 types of information nuggets that fit best relational aggregated search. They are *classes* (e.g. countries, French wines, US presidents), *instances*<sup>3</sup> (e.g. France, USA, Italy) and *attributes* (e.g. capi-

<sup>3</sup>The instance is a synonym for named entity. We prefer this term because it invokes a semantic relation with classes.

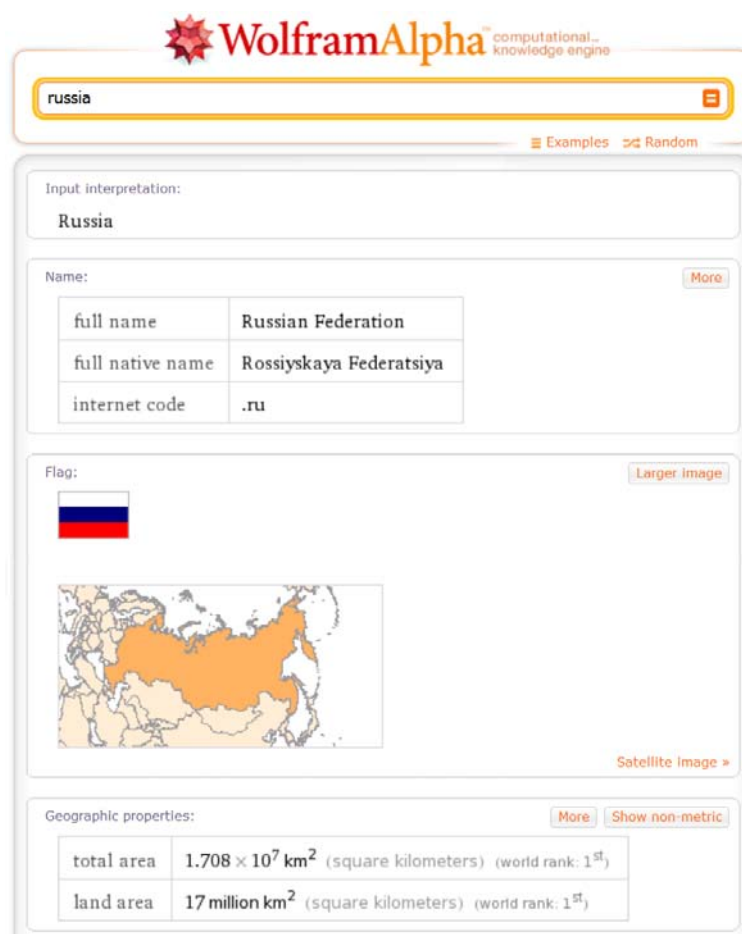


Figure 4.2: Wolfram Alpha result for “Russia” accessed on August 2011

tal:Paris, area:674,843  $\text{km}^2$ , demonym:French . . . ) [9].

**Relations:** In relational aggregated search, we are interested in all **nugget-nugget relations** that can be useful for information retrieval. Within these relations, we distinguish **instance-nugget** relations. These relations put in relation a given instance with whatever kind of content. This can be particularly useful, because we can at least answer queries about the instance with related nuggets. Here, we can place the relation between an instance and an image, video, document, definition. The relation can be “image  $X$  is an image of John Travolta”.

Within the different information nuggets, classes, instances and attributes are particularly interesting because for them relations are common and often implicit e.g. “capital:Paris” *is an attribute of the instance* “France” or “France” *is an instance of the class* “countries”. We can define here 3

specific relations:

- **instance-class relation:** This relation exists between the instance and its class e.g. “Everest”-“mountains”.
- **instance-instance relation:** This relation exists between two instances. It can be a connecting phrase such as “plays in” taken from the phrase “John Travolta plays in Pulp Fiction”. It can also be broader relations such as “has the same class as” or “relates to”.
- **instance-attribute relation:** This relation exists between the instance and its attributes e.g. “Everest”-“height”.

These relations have been addressed in Information Extraction (IE) and Natural Language Processing (NLP). Here extracting and relating happens mostly at the same time. We will detail more in section 4.4.

**Query dispatching:** Inspired by the work in [35], we distinguish three types of information needs where benefits from relational aggregated search are obvious. We call them *relational queries*:

- **attribute query** (“GDP of UK”, “address of Hotel Bellagio”)
- **instance query** (“Samsung Galaxy S”, “Scotland”, “Oscar Wilde”)
- **class query** (“Toshiba notebooks”, “British writers”)

For each of these queries we can trigger a different approach. From this perspective, query dispatching becomes important to detect the type of query.

**Result aggregation:** Relations can enable new ways to assemble search results. When the query is specifically asking for an attribute, the best choice can be returning its value right away. When the query is an instance, the best choice can be a summary of salient attributes (properties). When the query is a class of instances, the result can be a comparative table of the class instances with their attributes. Figure 4.3 shows what these results might look like.

Relational aggregated search has to deal with many open issues. First, we need to know which queries will benefit from relational aggregated search (see section 4.3). Second, we need to identify at large-scale (extract/acquire) relations between information nuggets (see section 4.4). Third, we need to transform extraction techniques into retrieval techniques with high recall (see section 4.5). Fourth, we have to assemble a final results based on the information nuggets and their relations (see section 4.6). In the next sections, we will detail on each of these issues with related work.

**Query:** *president of France*

<b>Result:</b>	Nicolas Sarkozy	
----------------	-----------------	--

---

**Query:** *Tower of Pisa*

<b>Result:</b>	<b>Location</b>	Italy
	<b>Width</b>	Tuscany
	<b>Height</b>	55.8m
	<b>Steps</b>	296
	...	

---

**Query:** *Macintosh notebooks*

<b>Result:</b>	<b>Mac Book</b>	<b>Mac Book Pro</b>
<b>Height</b>	1.08 inches	0.11-0.68 inches
<b>Width</b>	13.00 inches	11.8 inches
<b>Depth</b>	9.12 inches	7.56 inches
<b>Weight</b>	4.7 pounds	2.3 pounds
	...	

Figure 4.3: Examples of relational search results

### 4.3 Relational queries

To build relational aggregated search, we need to identify the queries that benefit the most from this paradigm. This is also important if we consider that different types of queries demand for different types of answers. In this section, we list queries from literature that fit in the relational aggregated search framework.

In [35], Cafarella et al. present one of the first query taxonomies for relational search. For these queries named entities and their relations find a crucial role.

- **qualified-list queries:** retrieve a list of objects that share multiple properties (e.g., west coast liberal arts college).
- **unnamed-item queries:** qualified-list queries that aim to locate a single object whose name the user does not know or cannot recall (e.g., the tallest inactive volcano in Africa).
- **relationship queries:** find the relationship(s) between two objects (e.g., the relationship between Bill Clinton and Justice Ginsberg).
- **tabular queries:** find a set of objects annotated by their salient properties (e.g., inventions annotated by their inventor and year of announcement).

In the above list, the query type and the type of result are binded i.e. the query definition includes the type of expected result.

Other interesting taxonomies of queries can be found in question answering [88]. In [121], authors list some queries that would benefit from focused retrieval and result aggregation. These queries include:

- **Factoid queries** (Who, what, when ...)
- **Definition queries** (What is ...)
- **Boolean queries** (expecting a yes/no answer)
- **Complex questions** (why and how questions)
- **List queries** (expecting a list of instances)

Although not all of the above queries can be answered directly with the relational framework, some of them can. List queries can be answered with a list of class instances. Definition queries demand for an explicit definition relation. Most of the factoid questions ask for named entities or their attributes although interpreting the factoid question is more a question answering task.

We will use a simpler taxonomy of queries inspired by [9]. The definition of queries is illustrated by examples and it is not binded with the type of expected result:

- *query by attribute* (“GDP of UK”, “address of Hotel Bellagio”)
- *query by instance* (“Samsung Galaxy S”, “Scotland”, “Oscar Wilde”)
- *query by class* (“Toshiba notebooks”, “British writers”)

Among all the presented queries, most of them benefit from relational aggregated search. We refer to the last taxonomy for the rest of the work, because it is simpler, it does not bind with the type of answer and it is closer to our conception of relational aggregated search.

## 4.4 How to acquire relations?

In this section, we list existing techniques to extract relations. First, we introduce generally these techniques from the perspective of information extraction. In the next sub-sections, we will take one relation type at a time and we will provide an overview on the existing techniques.

*Information extraction techniques correspond to rules that can be applied to sites, documents, or parts of documents to extract automatically classes, instances, attributes and their relations.*

Most of the extraction rules have the form of  $LxMyR$ , where  $x$  and  $y$  are meant to be two information extracts and  $L$ ,  $M$  and  $R$  are meant to



be patterns that are found respectively before, in between and after the two extracts. For instance, the rule “the  $x$  of  $y$  is” can be used to identify attributes of instances e.g. “the capital of France is Paris”. Rules that rely only lexicon (words, phrases) and part-of-speech tags are also referred to as *lexico-syntactic rules*, while rules that contain structure tags are usually known as *wrappers*.

Extraction rules can be hard-coded or learned. Hard-coded rules are usually intuitive rules which are easy to write. Machine learning is applied to learn new and complex rules. A survey on IE [41] classifies IE techniques in 4 classes with respect to the need for training (automation degree):

- **Hard-coded:** The extraction rule is not learned, but manually coded.
- **Supervised:** The extraction rule is learned from a labelled training set.
- **Semi-supervised:** The extraction rule is learned from a training set which is not labelled.
- **Unsupervised:** The extraction rule is learned without any training.

Information extraction techniques are quite heterogenous and they make use of various evidence such as statistics on terms, tags, decoration mark-up, part-of-speech tags, etc. This evidence is then combined to define rules that match classes, instances, attributes and their relations. We provide below just a short taxonomy of features (evidence) that are commonly used for this purpose:

- **word statistics [58, 7, 49, 10, 105, 71]:** Some terms are more frequent within or around information extracts. Statistics on words (often in conjunction with part-of speech tags) are helpful to learn common *lexico-syntactic patterns* for information extraction.
- **part-of-speech tags [57, 28]:** Information extracts are usually nouns or compound noun phrases surrounded by verbs, adjectives, prepositions, etc. Part-of-speech tags are helpful to learn possible patterns.
- **tags (HTML, XML) [49, 36, 96, 93, 97, 92]:** Most of the documents have some structure denoted through tags. The structure of the document is often useful to determine relations. In particular, HTML tables and HTML lists are known to contain relational data.
- **decoration, visual appearance [17, 117, 194]:** Sometimes the structure of a document is easier to learn through its visual aspects, especially when a pattern in terms of tags is difficult to define or learn.

- **external quality sources (Wikipedia, DBPedia) [164, 165, 188, 98, 105]:** Existing ontology-like and encyclopaedia-like sources provide already a large knowledge base which is used to learn patterns or to reinforce confidence on extraction.
- **PMI and search hits [44, 176, 141]:** Pointwise Mutual Information (PMI) [44, 176] is a statistical measure that indicates the relation between two expressions. An easy way to estimate PMI is through search hits which indicate the number of search results a search engine has to return on a given query [141]. PMI and similar statistical measures can play a crucial role to determine relations.

Most of the techniques in IE are domain-specific i.e. they are designed to work well for some classes or instances or attributes. Most of them are oriented towards precision. To enable relational aggregated search we need high recall and reasonable precision. From this perspective, domain-independent methods and high recall methods become crucial.

In the next section, we list approaches with respect to the relation type they target.

#### 4.4.1 Instance-class relation

The class instance is often referred in literature as *named entity*. Some of the pioneer research is met in the Message Understanding Conference [69]. Here, studies on named entities involved only 7 defined categories [69], namely person names, organizations, locations, date, time, money and percentage expressions, while today standardized taxonomies with 150 classes can be found such as the extended named entities hierarchy<sup>4</sup>. In reality, we cannot enumerate all possible named entity classes. A class can be as simple as “countries”, but it can also be “members of the UN Security Council” or “my friends”. Sometimes, we might not even be able to name the class in a reasonable way.

The definition of named entities as instances of some class makes the class-instance relation intrinsic for extraction techniques. These techniques are also known as *Named Entity Recognition* (NER). Initial work as mentioned is domain specific i.e. it addresses specific classes of instances. We can find approaches which are specific to locations [183], music [48], books [34] and so on.

Hearst [74, 75] proposes one of the pioneer domain-independent approaches to extract named entities and their classes. The author identifies 6 lexico-syntactic patterns which detect this relation. For instance the pattern “ $NP_0$  such as  $NP_1$ ” relates a class  $NP_0$  with one named entity  $NP_1$  ( $NP$  stands for noun phrase). We can illustrate with a sentence that matches

---

<sup>4</sup>The full hierarchy can be found in [http://nlp.cs.nyu.edu/ene/version6\\_1\\_0eng.html](http://nlp.cs.nyu.edu/ene/version6_1_0eng.html)

this pattern: “I have never visited big cities such as London”. We can also list some other patterns from this work: “ $NP$  such as  $\{NP, \}^* \{(and|or)\} NP$ ”, “ $NP \{, NP\}^*$  or other  $NP$ ”, “ $NP, \{, NP\}^*$  and other  $NP$ ”, etc.

In [71], Guo et al. study named entity recognition within queries. To identify named entities they use a probabilistic approach which relies on query logs and Latent Dirichlet Allocation. They represent the query as named entities and context (the part of the query which is not a named entity). Then, they train a classifier with triples of  $(i, c, cx)$  i.e. (instance, class, context). Their approach is experimented with 4 classes namely “Movie”, “Game”, “Book” and “Music” with an initial seed of 120 labeled triples. Their approach performs well on queries, but it is likely to lose performance if applied to longer text and to more classes.

In [58], Etzioni et al. present KnowItAll an Information Extraction system aiming domain-independent and unsupervised extraction. They integrate many of the state of the art lexico-syntactic rules from Hearts [74]. They also deal with subclass detection i.e. detect if class  $x$  is a subclass of class  $y$ . For instance, “chemist” is a subclass of “scientist”.

The class-instance relation is also targeted in TREC Entity Track 2009 [19] and 2010 [20]. One of the proposed tasks involves returning a list of named entities of a given class. In addition, one initial entity is given. The query can be similar to “Formula 1 drivers related to Luis Hamilton”. Here, we can find many pointers towards other approaches that we will not be mentioned here.

We can conclude that there is a large interest in identifying named entities and their classes. Existing techniques are promising, although they are mostly precision oriented.

#### 4.4.2 Instance-instance relation

Inspired from [164, 92], we distinguish four main relations that can relate an instance to another namely: *synonymy*, *sibling relation*, *meronymy* and *non-taxonomic relations*. To illustrate, “Big apple” is a synonym for “New York City”. France and Italy are siblings in that they are instances of the same class “countries”. Meronymy involves part-of-a-whole relations such as “Italy -is a member of- NATO”, “Quebec- is part of -Canada”, “CPU -is part of the- computer”. The non-taxonomic relations are relations between two instances given by textual description such as: “John Travolta -plays in- Pulp Fiction”, “Windows -is a product of- Microsoft”. The phrase “plays in” defines a relation between the instances “John Travolta” and “Pulp Fiction” as well as the phrase “is a product of” defines a relation between “Microsoft” and “Windows”. Non-taxonomic relations are usually defined through some text phrase. Typically, instances of one class share relations with instances of another class. For example, “Uma Thurman” plays in “Kill Bill”. The relation “plays in” is met between instances of “actors”

and instances of “movies”.

In [164], we are presented YAGO, a large extensible ontology built through careful combination of heuristics on Wordnet and Wikipedia. They extract synonyms, instance-class relations (hyponymy) and non-taxonomic relations. The result is a set of about 1 million instances and about 5 million relations. They focus only on some specific non-taxonomic relations such as “born in year”, “died in year”, “established in”, etc. Most of the heuristics are designed to obtain a high accuracy ontology.

Snowball [7] is another state of the art IE system which aims non-taxonomic relations between instances. Starting from a handful of training examples, it can learn new relations as well as many related instances. Training data and target data are represented as 5-tuples  $LxMyR$  which contain the instances and the text before, between and after them. Although the technique looks domain-independent, it depends on the training data i.e. another seed of training data can produce different results.

In the KnowItAll system [58], relations are learned starting from a seed of instances. Each instance is issued as a query to a Web search engine. The returned results are used to learn common relation patterns. Then, these relations are also used to learn new instances. In [57], Banko et al. propose TextRunner a system that runs an unsupervised domain-independent technique to extract instance-instance relations. The approach consists of a single-pass over data which extracts 3-tuples  $e_j r_{i,j} e_j$  (entity, candidate relation string, entity). Textual features and part-of-speech tags are combined in a Naive Bayes Classifier to learn relations. TextRunner is shown to achieve a 33% error reduction when compared to KnowItAll. TextRunner depends on the quality of the part-of-speech tagger and although it is designed to be domain-independent it works better for some classes.

The sibling relation is useful when we cannot associate an instance to its class. In this case, telling that two instances belong to the same class is enough. This is useful to identify instances with similar properties and for set expansion i.e. given a set of instances we enlarge this set with new instances of the same class [184, 129].

Instance-instance relations are mostly extracted through lexico-syntactic rules based on term statistics and part-of-speech tags. Some of these techniques are particularly interesting in that they are applicable at large scale and domain-independent. Nevertheless, some relations are difficult to capture as their extraction depends on the training data or the technique being used. So far, instances and their relations have been used few for retrieval purposes. In addition to massive acquisition of relations, research should investigate on ways to store and exploit them.

### 4.4.3 Instance-attribute relation

Attributes can find different uses. They can be used for the summarization (representation) of content [193, 125], to assist search such as for query suggestion [28] and faceted search [29], or for question answering [60]. Attribute acquisition methods can be domain-independent or domain-dependent. From domain-dependent approaches, we can mention approaches that focus on products. In this domain, attributes have been used to improve product search and recommendation [125, 141], but also to enable data mining [187].

To acquire attributes from the Web, it is common to use decoration markup [194, 187, 49] and text [28, 174, 135]. HTML tags (for tables, lists and emphasis) have also been shown to help for attribute acquisition [194, 187]. Wong et al. [187] combine tags and textual features in a Conditional Random Fields model to learn attribute extraction rules, but they need a seed of relevant documents manually fed.

Another common technique to acquire attributes is through the use of lexico-syntactic rules. For example, Pasca et al. [9, 136] use rules such as “A of I” and “I’s A” to acquire attributes from query logs. Authors represent the class as a set of instances and multiple class instances are used to improve extraction. In [10], authors use more precise lexico-syntactic rules such as “the A of I is”, but recall of these rules is lower. In [141], Popescu et al. use lexico-syntactic rules to extract product attributes from reviews.

At last, tables are known to be a mine for relational data and attributes. Cafarella et al. [37, 36] show we can identify billions of relational tables in the Web. They do not explicitly extract attributes from these tables, but they show how we can automatically classify tables into relational tables and tables with headers. This is an initial step towards massive extraction of attributes although we need additional processing to extract attributes and associate them with instances. In [43], Chen et al. identify attributes using column (or row) similarities. Another common technique to extract attribute from tables is through wrapper induction [41, 49, 186]. Given a training set or a set of similar documents, wrapper induction learns extraction rules. Many wrappers extract at record level, but they do not distinguish between attribute name and attribute value. Furthermore, wrappers are precision oriented and they work well only for some sites.

To summarize, current attribute acquisition techniques can obtain a high precision. Although many of these techniques produce a considerable number of attributes, they cannot cover the needs that can be answered with the Web. Most of them are conceived to work offline and they cannot extract instance attributes whatever the instance.

#### 4.4.4 Instance-nugget relations and nugget-nugget relations

We do not focus much on these relations as they are too broad and state of the art can explose. Nevertheless, it is worth mentioning that different relations have been identified and used differently in information retrieval.

The instance nugget relation can be whatever relation that relates an entity/instance to some content. The content can be an image, a document, a passage, a video and so on. For instance, in [170] Taneva et al. study image retrieval for instances. This is similar to assigning a relationship “image of”. In product search and opinion mining [141, 78], reviews are related to instances and their features (attributes). These relations are then used to summarize product reviews.

The nugget-nugget relation is the broader class of relations we define. We will list here just some illustrate examples without aiming to be exhaustive.

**Similarity (“similar to”):** Two information nuggets can be similar to each other. This is a very common relation. It can be found in clustering, classification, multi-document summarization, news aggregators, etc. For instance, news aggregators group similar news articles into stories.

**Diversity (“different to”):** On the other hand, the inverse of similarity represents another useful relation. Studies on novelty and diversity claim that it is better to promote some diversity among the retrieved results. This is useful to increase chances of guessing at least one relevant result [8], but also to find different aspects of the same information need [45]. Result diversification is today present in major search engines.

**Space-time (“happens in time/space”):** Content can also relate with respect to some features such as time and location. For instance, sometimes it is better to order information chronologically to favor freshness of information [55, 148]. In Geographic Information Retrieval the location feature is fundamental to organize search results [177, 82].

### 4.5 How to retrieve relations?

Information extraction is mostly an offline process which scans documents and labels information within them. In the context of search, we have to do with an online process which is query-dependent. We need to adapt information extraction techniques to assist search through online and high-recall extraction. We will call online relation extraction as *relation retrieval*. This means that given a query we want to return a scored list of relations.

Cafarella et al. [37, 36] perform relation retrieval using HTML tables in the Web. They return rows of tables that match a given query. Taneva et al.

[170] issue instance queries to retrieve related images. The above examples are just meant to illustrate that retrieving relations is sensed and possible. However, there is few work done to adapt retrieval for classes, instances and attributes.

In particular, we define *attribute retrieval* and *instance retrieval* as processes which will enable flexible and high recall answering for relational queries.

- *Attribute retrieval takes as input an instance (e.g. France, Homer, London) and returns a ranked list of attributes.*
- *Instance retrieval takes as input a class (e.g. countries, authors, cities) and returns a ranked list of instances.*

Existing work which targets attributes, instances, classes is mainly conceived as an offline process [41, 58, 7, 57, 28, 49, 164, 165].

Within attribute retrieval we can detach the attribute value retrieval problem. Indeed, if attribute retrieval identifies correctly the attribute name, we can launch a second retrieval process to identify its value/s. A separate treatment of attribute names and values is not novel in literature [194, 10]. If we treat attribute values separately, we can define another online process namely:

- *Attribute value retrieval takes as input an attribute name and an instance (e.g. (capital, France); (birthplace, Homer); (population, London)) and returns a ranked list of attributes values.*

These three relation retrieval approaches can all assist relational aggregated search in particular for attribute, instance and class queries. We believe that the most prioritary approach is attribute retrieval. Indeed, instance retrieval is handy mostly for class queries, while attribute retrieval (on names and values) can assist most types of relational queries. We will detail in the next section possible aggregation of the retrieved relations.

## 4.6 Result aggregation

This section is about ways to aggregate search results in the context of relational aggregated search. We will list related work and thoughts using one query type at a time.

Attribute queries (e.g. capital of France) should ideally be answered with the correct attribute value/s. However for many queries we might find many candidate values without being certain on their relevance and correctness. With respect to this issue, there is a lot of work in question answering

(what is the capital of Congo?) [88]. Here, the result is one or more candidate answers, typically associated with supporting text. Another mean to answer attribute question are knowledge bases (ontologies, encyclopaedia). In particular, we can mention DBPedia which allows querying through SPARQL its ontology to identify attribute values of some entity [16]. The user has to know SPARQL, the exact name of the entity and attribute name. Wolfram Alpha also answers attribute queries through its internal knowledge base. Major search engines have started to answer some attribute queries. They propose its value on top of the other search results. This is done rarely mostly when the answer is almost certain.

Instance queries can also be answered in many different ways. In object-level search, these queries are answered through records and attributes extracted from one or more pages [126, 125]. The answer can also be a set of related instances [24, 87], multimedia content [170], passages [155], attributes [93, 96, 97]. Specific approaches have been adapted for people search [110, 4], product search [126, 141, 5, 3], bibliographic search [80, 126, 2], etc. Most of the existing approaches are domain-dependent and some of them have low recall. In the relational framework, we are mostly interested at retrieving attributes at large-scale for whatever instance through high-recall and domain-independent techniques.

Class queries can be answered with a list of instances. This is the case for most approaches that can be found in Question Answering [88], TREC entity tracks [19, 20]. These queries can also be answered through knowledge bases such as Wikipedia and DBPedia which store class instance relations [16]. Google Maps<sup>5</sup> answers some class queries concerning geographic locations through a list of instances, addresses, rating and their visualization on a map. Google Squared answers these queries with a tabular result composed on the instances and their attributes. This is closer to the perspective of relational aggregated search.

We can conclude that there are several ways to answer queries in the relational framework. The quality of the result aggregation depends on the quality of the relation and the relevance of the result components.

## 4.7 Case studies

In this section, we propose two case studies. First, we analyze object-level search. The latter falls within the relational aggregated search, but its conception and terminology are slightly peculiar. Then, we present opinion mining as a special case where relations with instances and attributes are used to summarize content.

---

<sup>5</sup><http://maps.google.com>



### 4.7.1 Object-level search

In object-level search, the retrieval unit is not documents but objects [126]. An object is meant to be an entity with the corresponding related information, which is usually attributes and records [126, 101]. *From this point of view, generating an object demands information extraction and merging.*

It is often the case when the user is searching for an object (entity) e.g. Nokia e72, Restaurant Belvedere, London, Jim Belushi. In such cases, an object-oriented result might be preferable to lists of documents. A document might contain relevant and irrelevant content and often information about the same object can be sparse in several documents, while the object can be an aggregate of extracted information.

Objects are very common in Web data. They can be products [80, 125], books, academic papers [125] and so on. Objects can be described with a schema. For instance, in [118], the schema looks like:

- `<!ELEMENT Book (Title, PubDate, Price, Authors) >`
- `<!ELEMENT Title (#PCDATA) >`
- `<!ELEMENT PubDate (#PCDATA) >`
- `<!ELEMENT Price (#PCDATA) >`
- `<!ELEMENT Authors (Author+) >`
- `<!ELEMENT Author (#PCDATA) >`

The schema can be learned or known a priori [118, 72, 106].

Still, it is necessary to code or learn good wrappers for each object element. Such wrappers can be specific to some class or they can be class-independent (domain-independent).

Nie [125] identifies three common sub-tasks namely Object-level Information Extraction, Object Identification and Integration, Web object retrieval.

**Seed documents and IE:** Object-level search needs a set of seed documents to extract objects from. Typically focused crawlers are used to select useful resources depending on the domain (products, academic papers, etc.) [80, 106, 101]. Lin et al. [106] propose a meta-search based seed expansion in addition to focused crawling. Another alternative is the use of classifier to select pages that are useful with the respect to the type of object [126].

Information Extraction is vital to this domain. It is not only necessary to identify the objects and their categories, but also the related information, which is typically records and attributes [126]. For instance, for a product it is possible to extract its name, a description, a price, a production year and so on.

**Object identification and integration:** The same object can be found with different names as well as the same name can refer to different objects. It is necessary to disambiguate objects and integrate around the correct object all the related information.

Object-level search has to build quality structured objects. If a Web page is supposed to be coherent and comprehensible as it is created by humans, the object oriented result introduces a better focus, but also new issues. Reliability, completeness and ranking accuracy are some of these issues [126]. Objects can integrate information from different documents providing more complete answers, but the extracted information is error-prone. An attribute value correctness depends strongly on the IE techniques used as well as from the quality of the record and the document it comes from [125]. As well an attribute value can vary with time [128].

**Retrieval:** After assembling objects, it is necessary to rank aggregated objects with respect to the users information needs (queries). Lee et al. consider two important cases namely *personalization* which aims to maximize the satisfaction for a given user and *diversification* which aims to minimize the risk of dissatisfaction of varying user intents [101]. They propose to cluster object results to propose different user intents.

Typically, object-level vertical search focus on specific class of objects, but it is also possible to find search over multiple classes. For instance, Microsoft Academic Search provides search over authors, papers and conferences. Google Squared can also be seen as object-level search. It uses a simple result organization and it is domain independent (class-independent).

Object-level search represents an interesting example involving information extraction and result aggregation. The retrieved information is organized into object. Each object associates an entity to related information such as common attributes.

#### 4.7.2 Opinion mining

Textual information can be broadly classified into facts and opinions. With the advent of Web and the explosion of user-generated content, a lot of opinionated text is available today. We can find opinions in forums, blogs, discussion groups, user reviews and so on, where users share their point of view about entities and events.

Today, there is an increasing interest in mining opinions. It becomes important to detect opinion containing sentences and their polarity. In TREC Blog Track 2006-2009, several tasks have been proposed. They include the

opinion finding track (i.e. “What people think about  $X$ ”?) and polarity detection (i.e. “Find me positive or negative opinionated posts about  $X$ .”) [111].

In TAC 2008 Summarization Track [1], an opinion summarization task is proposed. The goal is to build short coherent summaries from answers to opinion related questions. For instance, one question can be “Why did people enjoy the movie “Good Night and Good Luck”?”.

Opinion mining can be applied to user reviews to automatically extract or summarize user opinions for specific entities such as for products, services [52, 159, 81]. Opinion summaries differ from text summaries. Opinions are associated (related) to entities, but they can also be associated to the features (attributes) of the entities [52]. Using relations between reviews and instances and attributes we can summarize and search more efficiently reviews across different sites [78].

For instance, in Wize.com [159] users can exploit user reviews from different sites for their search. The query includes the type of product and a selection criterion, such as “*best cameras for holidays*”. Products are ranked based on opinions about the product and the specified criteria. These opinions are extracted from different sites of user reviews.

Opinion mining relates to relational aggregated search. Information extracts, relations and opinion mining are combined to provide a sentiment or a summary of sentiments.

## 4.8 Conclusions

In this chapter, we presented relational aggregated search as a distinct instance of aggregated search. Here, information nuggets can be decomposed and related to each other. In particular we focused on classes, instances, attributes and their relations. We presented the existing techniques to acquire them and we propose possible result aggregation for these queries. Although existing techniques can extract many classes, instances and attributes, an additional effort is needed to increase the recall of these methods for search applications. In other terms existing techniques are designed to work offline and they are experimented in an offline setup. They rely on precise patterns and they have not been proven to scale for search applications.

In particular, we identified in recall-oriented techniques such as attribute retrieval the potential to enable relational aggregated search. The goal is to retrieve attributes (names and values) for given queries (instances, classes) which can by analogy be generalized to other approaches such as instance retrieval, passage retrieval, etc. The retrieved content and relations can then be used to assemble relational aggregated search results.



## Chapter 5

# Cross-vertical aggregated search

### 5.1 Introduction

*Traditional Web search* returns a list of snippets of Web pages. Queries such as “cheap flight from London to Paris” and “photos of Eiffel Tower” will not be answered directly from this approach. The user has to scroll the list of snippets and scan within Web pages to satisfy his information need. By contrast, a vertical search engine for traveling can provide a more accurate result for the first query and an image search engine can answer right away the second query.

Nowadays, it is possible to access results from some of the vertical search engines directly from the Web search interface, which is already the case for the major search engines such as Google, Yahoo! Search and Bing. This approach provides great visibility to vertical search engines, while it enables Web search to integrate the advantages of vertical search engines. *We define as cross-vertical aggregated search (cvAS) the task of searching and assembling information from vertical search engines and Web search to distinguish with federated search and meta-search.*

One way to aggregate vertical searches is to place content of the same type into predefined layout panels. This is the approach chosen by Yahoo! Alpha<sup>1</sup> (see image 5.2), Ask<sup>2</sup> and Kosmix<sup>3</sup>. Whenever a minimal amount of vertical content is available for a query, the predefined layout panel is used. Major search engines have tried other approaches. They started integrating vertical content only on top or bottom of their search results list (see figure 5.3), but nowadays, they rank results by blocks. Precisely, groups (blocks) of results from each source are ranked with each other and assembled in

---

<sup>1</sup><http://au.alpha.yahoo.com>

<sup>2</sup><http://www.ask.com>

<sup>3</sup><http://www.kosmix.com>

the same panel. Within each block, results are usually ranked i.e. they are showed in the order they are returned from the origin source (see figure 5.1). For instance, we can have a block of three Web search results followed by a block of images, followed by a block of news search results and then Web search results again. The advantage of this approach is that the user views results in an ordered and uniform way. We can also integrate new sources without needing more visualization space for search results (e.g. additional panel).

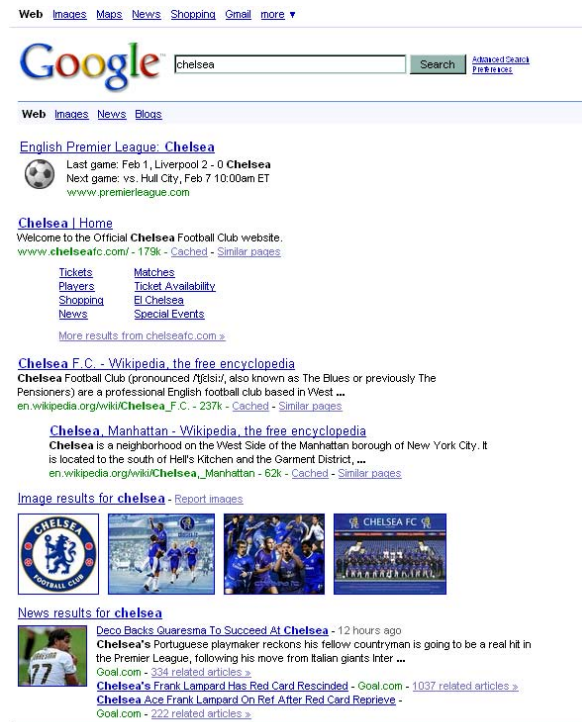


Figure 5.1: Google Universal search results on the query “Chelsea”, accessed on April 2009

In cross-vertical aggregated search it is easy to identify the components of our general framework for aggregated search (query dispatching, nugget retrieval and result aggregation). Query dispatching will correspond to the selection of sources. Each source will perform its nugget retrieval process and finally retrieved nuggets will have to be assembled in one interface. Although the tasks are well distributed, the problem is far away from being solved. It is not easy to decide which sources should be used and how the retrieved results should be assembled and presented. In particular, we can list some major issues which have the attention of current research:

- **Source selection and representation** Which source should be used?

How should they be represented internally in terms of features? [14, 104, 53, 99]

- **Result aggregation:** How should search results from different sources be assembled (ranking by item, ranking by block, ...)? [139, 12, 108]
- **Result presentation:** Which are adequate interfaces for cross-vertical aggregated search? [166, 168] (presentation)
- **Evaluation:** Which are the advantages of aggregated search and how can they be assessed? [94, 13, 196]

In the next section, we list some of the advantages of cross-vertical aggregated search. Then, we will focus on the research issues one by one.

## 5.2 Advantages of cross-vertical aggregated search

Before cross-vertical aggregated search was implemented by major aggregated search engines, different studies [108, 14] on query logs have shown that vertical search intent was often present among Web search queries. For example, queries such as “Avatar trailer” or “Jennifer Lopez images” can be found in Web search logs, although these queries might have been issued for videos and images.

Liu et al. [108] analyzed 2153 generic Web queries into verticals, using query logs. They found that 12.3% have an image search intent, 8.5% have a video search intent and so on. Arguello et al. [14] classified 25195 unique queries, randomly sampled from search engine logs, into 18 verticals. 26% of the queries (mostly navigational) were assigned no vertical, 44% of the queries were assigned one vertical and the rest of the queries were assigned more than one vertical. The latter were mostly ambiguous.

Sushmita et al. [166] analyze the advantages of cross-vertical aggregated presentation of results in comparison with the traditional tabbed access of sources. They used 6 broad tasks as topics and they observed an increase in the diversity and number of relevant results accessed by the users. A similar approach is used in [158]. Image, news and Web results are used in conjunction with clustering. This is done to increase the user’s result space.

Some of the advantages in cross-vertical aggregated search should be sought in the diversification of search results. The importance of result diversification is known since earlier times. Goffman states that a document should not only be relevant, but also novel [63, 39]. Diversification increases chances to guess at least one relevant result [8] and it can help users to collect multiple aspects of an information need [45]. In [154], Santos et al. analyze ambiguity across 30 queries which have been met in query logs across 4 different sources namely Web search, image search, news search and product search. They could identify several interpretations (relevant

aspects) for each query. Each source could satisfy more than one ambiguous interpretation. Nevertheless, some interpretations were found more frequent than others. We can confirm from this study that cross-vertical aggregated search helps to deal with ambiguous queries.

The facts that vertical intent is often present within Web search queries and that cross-vertical aggregated search produces a diverse set of relevant results proves only partially the usefulness of cross-vertical aggregated search. The vertical search engines should prove their aptitude to answer well and frequently queries from the embedding source (usually Web search) as well as the embedding source should integrate appropriately these results to improve its effectiveness. Furthermore, we need to know when and why two different sources can be relevant at the same time. Do they contribute to answer the same information need. Do they provide redundant information? Are they complementary to each other?

### 5.3 Source selection and representation

Some aggregated search systems make use of all sources they dispose for all queries, while others are *source selective* i.e. they make use only of the sources considered useful for the query. Because, most of the major current approaches are source selective, deciding which source is useful (relevant) becomes one of the primary issues in cross-vertical aggregated search. This corresponds in research to the *vertical selection* problem [14], which we will call as *source selection* to enable generalization on Web search and vertical search. *Source selection consists in predicting whether a source is relevant or not for a given query/information need.*

Typically, source selection aims avoiding delay times that would be caused if many sources are inquired and we wait for all results. Though, sources are typically selected before retrieving results. To enable efficient selection, sources have some internal representation in the system. This representation can be as simple as a textual description of the source, but in general it contains representative terms and features for the source extracted from sampling or other mining techniques. We will list some of these techniques below starting from the perspective of federated search.

The source selection and source representation problems are first met in federated search. Here, we meet the term resource as synonym of source. In federated search, it is common to distinguish between cooperative sources and non cooperative sources. In the case of cooperative sources we can access the collection of each source and though compute useful statistical data on the collection terms. But, in federated search sources are mostly non cooperative, though we cannot access the entire collection. In the latter case, the source representation can be a manually written description of the source [65], although this is generally not enough. Typically, a sample of



representative documents is obtained issuing queries to the source. There are two main approaches. In the first approach [38], an initial representation is built using top search results from one seed query. The representation is then used to generate new queries and update the representation. Alternatively, Shokohui et al. [160] show that better results are obtained when most frequent queries are used to generate the sample. In the case of cooperative sources, we can use all collection to select a set of representative terms and features for the source.

In cross-vertical aggregated search, there exist approaches where the source is selected after retrieval i.e. given some of the search results the source has to offer, it is decided whether the source is useful or not. Though, when we speak of internal representation of sources in terms of features, we distinguish between *pre-retrieval features* and *post-retrieval features*. All these features are also presumed to be useful for ranking and assembling results.

Table 5.1 contains a list of common pre-retrieval features used in literature listed with the work they appear in. In [12, 139, 14], authors rely on terms that indicate vertical intent. Some of them are obvious and they can be input manually such as the terms “photo”, “image”, “video”, “clip”. Others are derived from the source representation or external sources. In [14] they propose mapping verticals to Wikipedia articles which tend to be uniform and verbose. For instance, a vertical search engine on autos can be mapped into the Wikipedia articles of the “Automobile”, “Car” and “Vehicle” category. This is particularly useful for sources that contain few textual data.

Queries are also associated to predefined categories such as “sports”, “arts”, “technology”. This is done in [139, 12, 167]. In [12], authors map queries to 30 topical categories derived from the Open Directory Project (ODP). Another important source of evidence is found in query logs of the sources. If a query has already been issued in one of the verticals from the source interface itself, this is a strong indicator of vertical intent. This approach is used in [12, 14].

Table 5.2 contains a list of common post-retrieval features used in literature listed with the work they appear in. This set of features has been shown to be very useful to score sources and blocks of results [12]. In the case of cooperative sources, we can have access to the relevance scores of the source itself or the source confidence on the utility of these results [139]. Nevertheless, in general we need to compute features that are uniform across sources in order for scores to be comparable. Ponnuswami et al. [139] compute the BM25 weighting function, while Arguello et al. [12] combine 4 different scores: (1) the cosine similarity between the query and the document/title representation, (2) the maximum number of query terms appearing consecutively in the document/title representation, (3) the percentage of query terms appearing in the document/title representation (4) the percentage of

the document/title representation that matches query terms.

The number of returned results from a source can also be source of evidence. For some sources, the freshness of the returned results is more indicative. For instance, microblogs are usually preferred when they are issued recently.

Source selection approaches correspond generally to supervised classifiers [104, 53, 14] that are trained through 3-tuples (query, source, relevance). The relevance assessments have been collected through human assessments on queries and intents [14] and through implicit feedback derived from click-through analysis [53]. These techniques will be described in detail in the next section.

Table 5.1: Pre-retrieval features

Feature based on	Description
Vertical intent terms [12, 139, 14]	Some terms indicate vertical intent such as image, video, photo. Some others can be related. This feature combines hard-coded and learned association rules for queries and sources
Query logs [12, 14]	These features indicate if the query has been met in a source query log.
Recent popularity of the query [53]	This feature indicates how often the query has been met in a source query log recently.
Click-through [12, 139, 104]	These features are generated from the documents that have been clicked for the query. The click is considered implicit feedback.
Query domain [14]	This feature is generated through classification of the query into predefined domains such as sport, arts, technology.
IsNavigational [139]	This feature indicates the chances of the query to come from navigational needs.
Query length [139]	This feature corresponds to the length of the query in terms.
Relevance feedback [54]	This feature is computed based on explicit and implicit feedback on the query and its intent.
Named-entity type [12]	These features indicate the presence of named entities of some type in the query.

## 5.4 Result aggregation

There are different ways to assemble results in cross-vertical aggregated search. Some approaches rely on source selection [104, 53, 14]. They integrate search results from a source on top of Web search results only when the

Table 5.2: Post-retrieval features

Feature based on	Description
Vertical relevance score [53]	This feature corresponds to the relevance score of the vertical source itself for a result or a block of results.
Query-results match [12, 139, 104]	These features correspond to match score computed on one or more search results from the source. This score can be returned from the source itself or computed from scratch on the result.
Number of results [12]	This feature corresponds to the results count of a source.
Freshness of documents [12, 53]	This feature indicates how fresh are search results for the query within a source.
Contextual score of results	These features indicate the relatedness of search results with respect to some context.
Geographic-context score of results	These features indicate the relatedness of search results with respect to some geographic context.

source is deemed relevant. Some others go beyond source selection. They rank results with each other. There are two main approaches in this direction. Some rank results in blocks of results of the same source [139, 12] and others rank results one by one [108].

Diaz studies the integration of news search results within Web search [53]. They estimate *newsworthiness* of a query to decide whether to introduce news results on top of the Web search results. They make use of click-through feedback to recover from system errors. Liu et al. [104] also rely on click-through data to extract implicit feedback for source selection. They represent queries and documents as a bipartite graph and they propagate implicit feedback in this graph. Their approach is experimented for the integration of product search and job search results.

Arguello et al. [14] list various sources of evidence that can be used to tackle source selection such as query-log features, vertical intent terms, corpus features. In later work [54], they show how they can integrate implicit and explicit feedback for vertical selection. From a set of 25195 labeled queries, they propagate implicit feedback across about ten million queries. In [15], the same authors show how to adapt source selection methods to new unlabeled verticals.

In [139] and [12], search results are ranked in blocks of vertical content (e.g. 3 images versus 3 Web search results). Ponnuswami et al. [139] use as training data pairwise preferences between a block of Web search results and a block of vertical search results. Similarly, Arguello et al. [12] also rely on pairwise preferences to train their ranking functions. They experiment

three ranking techniques: one derived from classification, a voting approach and learning to rank techniques. Learning to rank techniques are shown to work best.

In [108], Liu et al. define a probabilistic model that enables ranking search results from different sources. In contrast with other approaches, they rank single items (search results) instead of blocks of search results. Although the probability model is interesting, the probability estimates are not convincing.

Current result aggregation approaches are inspired from the approaches taken by major search engines. We need a more flexible framework for result aggregation which enabled different ways to put results together and preferably less training or no training.

## 5.5 Result presentation

In this section, we focus on result presentation. We will first list some illustrative examples from existing commercial applications. Then, we will classify and analyze approaches.

One way to aggregate vertical searches is to place content of the same type into predefined layout panels. This is the approach chosen by Yahoo! Alpha<sup>4</sup> (see image 5.2), Ask<sup>5</sup> and Kosmix<sup>6</sup>. Whenever a minimal amount of vertical content is available for a query, the various predefined holes are filled and displayed around the natural search listings. The advantage of such an approach is that the user knows where the vertical content is shown. If he/she knows what he is looking for, he/she knows where to expect the relevant content, while it remains possible to benefit from all sources. On the other hand, vertical content is almost always shown even if there is nothing relevant for such a type of content. As well, this approach limits the number of vertical searches that can be integrated at once.

Some of the major search engines started integrating vertical content only on top or bottom of their search results list. This is the case for Microsoft's search engine Bing (in 2009) (see figure 5.3) in its initial times, but it was also the case for Yahoo! Search and Google some time ago. In this case, web search results are ranked as usual. When it is probable that the user is looking for other types of content such as images, news, video, this content is placed in the bottom or the top of the Web results list. One of the advantages is that vertical search results are added only when needed and visualization remains simple. However, this approach enables integrating search results from just one or two vertical search engines.

---

<sup>4</sup><http://au.alpha.yahoo.com>

<sup>5</sup><http://www.ask.com>

<sup>6</sup><http://www.kosmix.com>

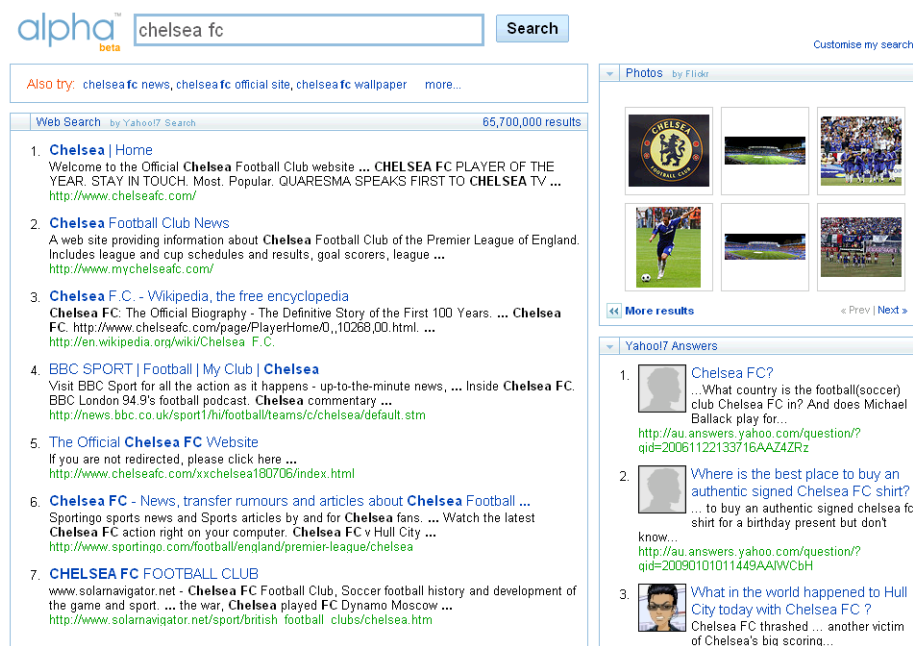


Figure 5.2: Yahoo! Alpha search results on the query “Chelsea FC”, accessed on April 2009

Nowadays, major search engines rank results by blocks, i.e groups (blocks) of results from each source are ranked with each other. Within the block, results are usually ranked i.e. they are showed in the order they are returned from the origin source (see figure 5.1). For instance, we can have a block of three Web search results followed by a block of images, followed by a block of news search results and then Web search results again. The advantage of this approach is that it is simple and at the same time flexible. We can add as many vertical searches as needed and to show only the ones which are more relevant.

In the Web, we can also find other picturesque applications such as Spezify<sup>7</sup> which proposes an original way to exploit visualization space (see figure 5.4). Spezify aggregates results from different vertical sources with a slight preference for content with immediate visual impact such as images, videos. Each result fills a rectangle, while different rectangles are placed side to side to fill the visualization space. The results expand in all direction (up, down, left, right) and it looks as a mosaic filled of content. The user gets an immediate image of the content and he can scroll it in all directions. Google

<sup>7</sup><http://www.spezify.com>

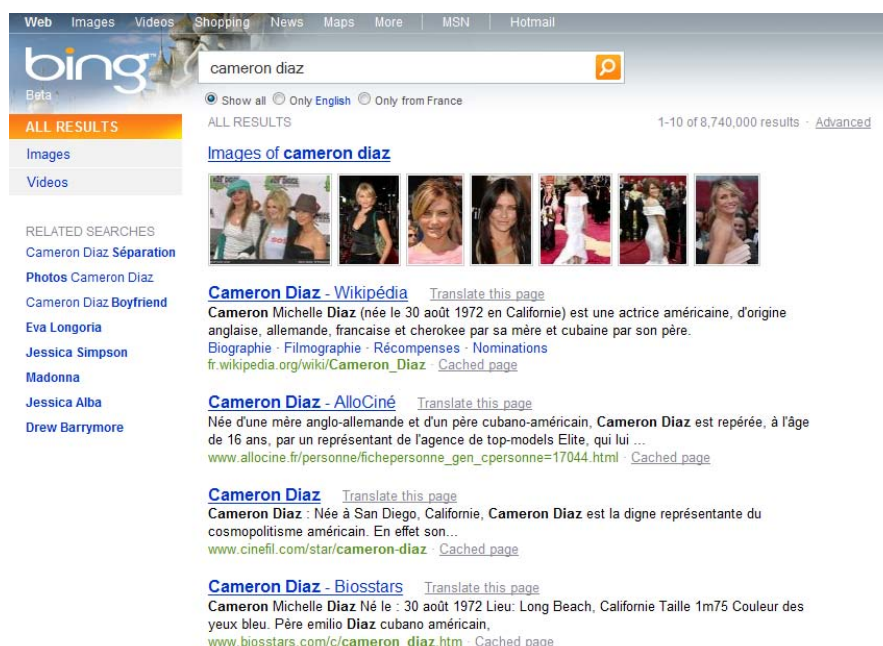


Figure 5.3: Bing search results on the query “Cameron Diaz”, accessed in april 2009

has also launched a new application called “what do you love”<sup>8</sup> (figure 5.5). This application aggregates results from several verticals as well as from other Google applications such as Google Translator, search term popularity measures and so on. The user of this application can save the returned results for future use or share them with friends.

The above examples were meant to show that there exist different possible and reasonable ways to assemble search results in cross-vertical aggregated search. We will now analyze them with respect to existing work in research. Here, we can classify approaches in two broad classes namely blended result aggregation and unblended result aggregation [166, 108].

**Blended result aggregation:** consists of merging search results from different sources with each other into the same panel.

**Unblended result aggregation** aggregation consists of keeping search results from different sources separate. Results are shown in different panels.

The “Yahoo! Alpha”-like and “Bing (year 2009)”-like approaches are examples of the unblended approach, while the Google-like and Spezify-like approaches use the blended approach. This classification might not be

<sup>8</sup><http://www.wdyl.com>

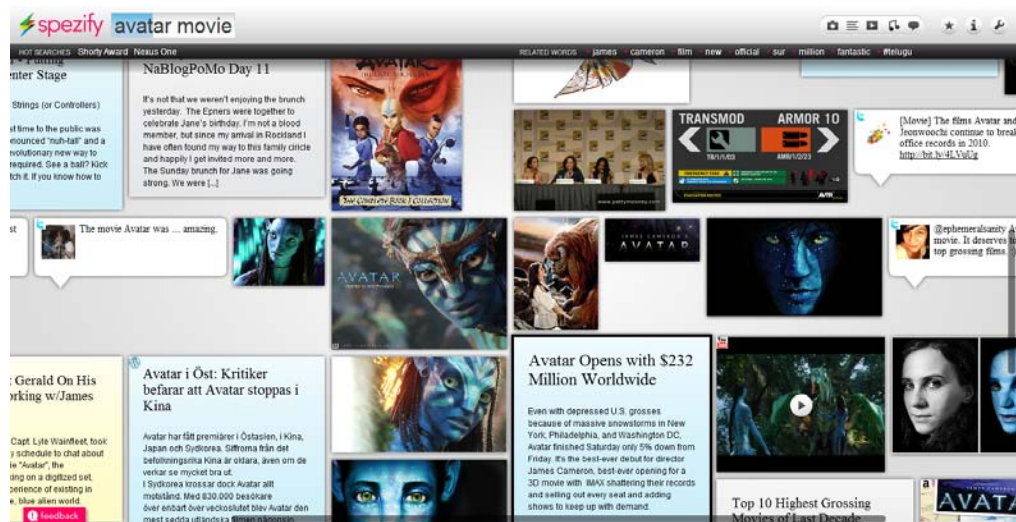


Figure 5.4: Spezify search results on the query “Avatar movie”, accessed on June 2010

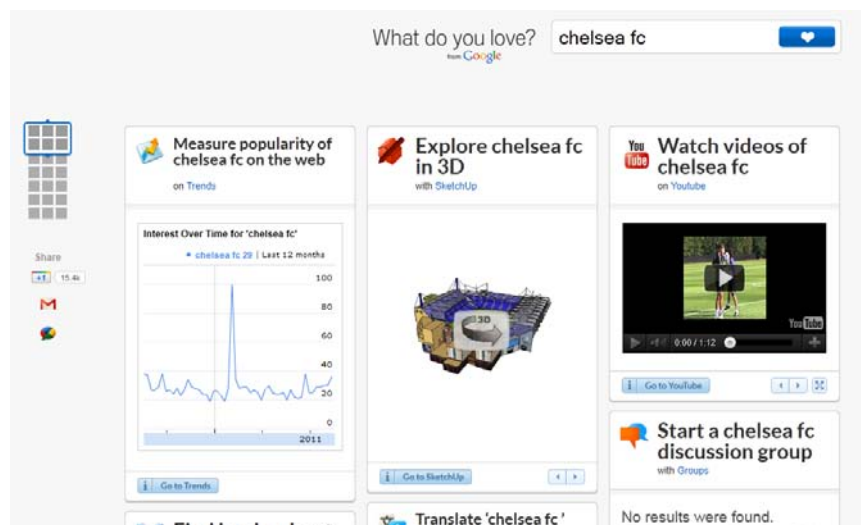


Figure 5.5: “What do you love” search results on the query “Chelsea FC”, accessed on August 2011

enough. Yahoo! Alpha and Bing (year 2009) do not blend results, but they are clearly different. They both place content into predefined areas in the visualization layout, but in the first approach the content is placed if available and in the second case if probable. We can say that the first approach is not source selective, while the second is source selective.

## 5.6 Evaluation

Until now, different evaluation methodologies have been undertaken for measuring the effectiveness of cross-vertical aggregated search. Because existing techniques have been designed with different goals, they are quite heterogeneous. We can classify them with respect to their target. In [14, 104, 108], the main goal is to evaluate source selection. In [166, 168, 173], the main goal is to compare cross-vertical aggregated search interfaces. In [13], authors target result ranking. We will describe here the different evaluation approaches.

One common protocol to evaluate source selection is to ask human participants to choose which are the relevant sources for a query [14, 104, 108]. Liu et al. [108] performed this kind of relevance assessment on 2153 generic Web queries. In [14], human judges classified 25195 queries. This kind of assessments is fast, but not necessarily accurate. The human judge might not guess the real information need or might neglect some interpretations of the query and some queries might demand specific knowledge.

In [166, 168], Sushmita et al. compare the effectiveness of different interfaces for cross-vertical aggregated search. They show that users find more relevant results when vertical results are placed together with Web results. They also show that placing vertical results on top, on bottom or in the middle of search results can affect the amount of vertical search results accessed by users. In both studies, participants are shown concrete search results from the considered sources. They are also given the information need behind the query. They have to click on results and bookmark the ones that are relevant. This approach is closer to traditional IR evaluation. The information need is not ambiguous and assessors can access real search results.

Instead of human participants, relevance assessments have been simulated using click-through logs [167, 53, 168]. In [53], Diaz shows that queries which obtain a high click through rate within news results are probable to be newsworthy. Click-through logs are also used in [168]. It is shown that for some sources such as video click-through behavior is different. Although click-through logs enable a large scale automatic evaluation, they cannot be as realistic as human based evaluation.

Recently, Arguello et al. [13] proposed a methodology to evaluate result ranking. Relevance assessments are pair-wise preferences between result sets. Each result set contains results from one source. This work does not focus on the notion of source relevance, rather than on the relative effectiveness of ranking.

Zhou et al. [196] propose building an evaluation benchmark for cross-vertical aggregated search through the re-use of existing evaluation benchmarks. They make use the the CluWeb track in TREC [46]. They artificially build vertical collections using classification. Then, they choose topics that



cover many verticals. We can see this work as a step towards evaluation benchmarks, although a more substantial effort is needed in this direction to make the distribution of topics, sources and assessments more realistic.

In general, evaluation of cross-vertical aggregated search remains an open problem. There are different types of relevance assessment, different measures, while there is no common agreement yet. In particular, it is not clear which are the advantages of this approach that are priority and how they should they be assessed. We know that cross-vertical aggregated search can provide focus and diversity, but we do not know why and at which extent this improves information retrieval. Research needs to investigate more on the interest and evaluation of cross-vertical aggregated search.

## 5.7 Conclusions

In this chapter we provided an overview on the cross-vertical aggregated search. We have organized related work around some main issues: source selection, result aggregation, result presentation and evaluation. Some of the related work is inspired from federated search while some is quite novel. Source selection has to take into account for vertical search and web search specificities. Result aggregation and presentation approaches go beyond the uniform ranked list. Evaluation techniques are also specific to the targeted issue (e.g. source selection, result aggregation, result presentation).

However, research in this direction remains quite heterogeneous and should be taken to converge. We need to clearly identify the novel issues and advantages of cross-vertical aggregated search. Which are the advantages of combining diverse sources remains to be explored. As well, we have seen that there are different ways to assemble and present results, which can affect user satisfaction. Existing evaluation techniques rely on binary relevance assessments but different setups. To better capture the utility of cross-vertical aggregated search we need to investigate more on the notion of relevance for cross-vertical aggregated search and the ways to capture it realistically. This will also be the goal of our contribution in this research direction.

Moreover, vertical search engine results can be combined in other tasks and uses except of Web search. We believe that the potential of this domain is to discover and research outcome will be proliferous in future years.



## Part III

### Part 3: Relational aggregated search: Attribute retrieval, result aggregation, instance lists and applications



## Chapter 6

# Attribute retrieval

### 6.1 Introduction

Semi-structured HTML data (tables and lists) in the Web are probably the largest source of relational data in the Web [37]. Because, we target high-recall for relational aggregated search in both terms of queries that can be answered and information that can be retrieved we propose approaches that rely on this kind of data. To do so, we do not rely on patterns for text extraction [9, 58] nor on precise wrapper induction techniques [41]. Our work is mostly inspired from the work of Cafarella et al. [37]. Their work is at our knowledge the largest mining from HTML tables and HTML lists for search purposes. They show how to identify relational tables and lists, although they do not specifically extract attributes or instances from nor they perform search for them. Our work is different because we explicitly extract and we perform query matching i.e. we rank by relevance.

In this chapter, we propose an approach for attribute retrieval i.e. given a query (instance or set of instances) our approach can extract candidate attributes and rank them by relevance to the query. We will show in the next chapter how they can be used to build aggregated information retrieval answers. In chapter 8 we present our research on HTML lists to extract sets of instances. In chapter 9, we apply our research to build relational aggregated search prototypes.

As we mentioned, for our attribute retrieval approach we rely on relational HTML tables. Although tables are probably the largest source of relational data [37], the task is not easy for the following reasons. Many of the HTML tables are not in a relational form, especially the ones used for layout design and navigation. Some tables are useful but not uniformly relational (see table 2 in figure 6.1) As well, relevant tables are not easy to detect. A table can be relevant for an instance even when the instance is not present within the table text.

Our approach takes into account for the above issues and it is designed

to work at large scale with high recall. It is composed of three main steps:

- retrieval of a seed of potentially relevant tables
- filtering of useless tables and attributes
- ranking of attributes by relevance features

To test our approach, we use three search situations. First, we retrieve relevant attributes for one given instance (e.g. “University of Strathclyde”). Second, we retrieve attributes for a given class (e.g. “universities”) represented as a set of instances. Third, we retrieve attributes for one instance when some other similar instances are given (from the same class). We use the term *instance attribute retrieval* when we retrieve attributes for one instance. By analogy, we use the term *class attribute retrieval* when we retrieve attributes for a class. The last search situation corresponds to reinforced instance attribute retrieval. However, the instance attribute retrieval is the core problem in all cases.

Moreover, we compare attribute retrieval from HTML tables with other existing approaches from state of the art. Concretely, we use the lexico-syntactic rules used in [9] and a combination of DBpedia and Wikipedia.

In the next section, we describe our general approach. Then we describe our experimental setup (section 6.3) and the corresponding results (section 6.4).

## 6.2 Attribute retrieval

The core of the attribute retrieval problem is to extract and rank attributes with respect to an instance query  $i$ . If we are able to do so, we can also retrieve attribute for a class (given as a set of instances). Here, we describe our approach composed of three main steps:

- **Retrieval of a seed of potentially relevant tables:** The first step corresponds to the collection of potentially useful relevant tables. Instead of retrieving only tables that match the query, we retrieve all tables that are within documents that match the query. This is done to avoid an important loss of recall. However, the problem is far away from being solved. Tables in the Web are quite heterogeneous and many of the retrieved tables are partially or not relevant.
- **Filtering of useless tables and attributes:** The second step corresponds to a recall-oriented filtering over tables and attribute lines. We apply three filters on tables and attributes namely *relational filter*, *header filter*, and *attribute line filter*. They are recall-oriented classifiers that can filter out many useless tables and useless attribute lines.

The first two filters target tables that are not relational and that do not have headers. The remainder of these tables is then used to extract attribute lines (rows or columns), which are then input to the attribute line filter to remove non conform candidates. This step is necessary because we can a last chance to detect useless data when we check at line level.

- **Ranking of attributes by relevance features:** The extracted attribute lines are ranked with respect to their relevance. The task is not easy. If we consider only tables that match the instance, we would lose many relevant tables (e.g. table 2 in figure 6.1 is relevant for “France”, but does not match it). Extracted attributes are ranked with a relevance score  $\phi(a, i)$  which is applied over an instance  $i$  and an attribute  $a$ . It will be described in details in section 6.2.2.

This approach is easy to generalize to class attribute retrieval. Here the query is a set of instances  $I$  that represents a class. For example the class “countries” can be represented by the set: “France”, “Italy”, “UK”, etc. The relevance score for an attribute  $a$  for the class of instances  $I$  will be:

$$\phi(a, I) = \frac{\sum_{i \in I} \phi(a, i)}{|I|} \quad (6.1)$$

In other terms, a relevant attribute for the class is likely to be present in many instances of the class. Though, we average the relevance score across all instances.

As well, we can easily deal with reinforced instance attribute retrieval. Let  $i$  be the target instance and let  $I_+$  be a set of similar instances given to reinforce retrieval. The hypothesis is that having more instances can improve attribute retrieval due to the simple fact that similar instances share common attributes. However, even if most relevant attributes are shared among instances of the same class, many exceptions can occur. For example, “queen” is relevant for the instance “UK”, but it is irrelevant for the instance “USA”. To overcome this problem, when it comes to ranking, we privilege attributes retrieved for the instance  $i$  using only the instances in  $I_+$  to reinforce ranking. The relevance score for the instance  $i$  will be computed as follows:

$$\phi'(a, i) = \begin{cases} 0 & \text{if } \phi(a, i) = 0 \\ \phi(a, i) + \phi(a, I_+) & \text{otherwise} \end{cases} \quad (6.2)$$

Due to the complexity of the problem, we evaluate separately the performance on attribute names and attribute values. Let the query be the instance “France”. The attributes names “capital”, “president”, “area”, “population”, “GDP”, “GDP per capita” are examples of relevant attribute

Relational Filter	Header Filter
$f_1$ : # rows	$f_1$ : # rows
$f_2$ : # cols	$f_2$ : # cols
$f_3$ : %rows w/mostly NULLS	$f_8$ : %head row cells with lower-case
$f_4$ : # cols w/non-string data	$f_9$ : % head row cells with punctuation
$f_5$ : cell strlen avg $\mu$	$f_{10}$ : % head row cells with non string data
$f_6$ : cell strlen stddev $\sigma$	$f_{11}$ : % cols w/non-string data in <i>body</i>
$f_7$ : $\frac{\sigma}{\mu}$	$f_{12}$ : % cols w/ $ len(row\_1) - \mu  > 2\sigma$
	$f_{13}$ : % cols w/ $ \sigma < len(row\_1) - \mu  \leq 2\sigma$
	$f_{14}$ : % cols w/ $ \sigma > len(row\_1) - \mu $

Table 6.1: Features for the relational and header filter

names. The attribute values can be retrieved with errors even when the attribute name is correct. For example, table 1 in figure 6.1 has relevant attribute names for the instance “Germany” although none of the values will be relevant/correct.

			Facts				
	Population	Median age	Capital	Paris	1	Loud	Rihanna
France	64,768,389	39.7 years	Demonym	French	2	The king of limbs	Radiohead
Italy	58,090,681	43.7 years	Population		3	Femme fatale	Britney Spears
Table 1			Population	64,768,389	Table 3		
			Median age	39.7 years	Table 2		

Figure 6.1: Examples of interesting tables

In the next sections, we will explain how our filters (*relational*, *header* and *attribute line*) work and which are relevance features used to compute  $\phi(a, i)$ .

### 6.2.1 Filters

#### Relational tables and attribute headers

We build the relational filter and the header filter using the same features as done by Cafarella et al. in [37]. Features on the left of table 6.1 are used to learn a rule-based classifier for the relational filter and features on the right are used to learn a rule-based classifier for the header filter. Learning is done with a training set of human-classified tables described later in the experimental setup.

Features include the dimensions of the table ( $f_1$ ,  $f_2$ ), the fraction of lines with mostly nulls (empty cells) ( $f_3$ ), the lines with non-string data



(numbers, dates) ( $f_4$ ), statistics on the character length of cells and their standard variance ( $f_5, f_6, f_7$ ), conformity of the header cells (lower-case, punctuation, non-string data) ( $f_8$ - $f_{11}$ ) and different statistics on the column lines with respect to their cell lengths (how many (%) of these lengths do not follow normal distribution) ( $f_{12}, f_{13}, f_{14}$ ).

We have to consider that relational tables can be oriented vertically or horizontally depending on the orientation of attribute lines. For example, tables 1 and 3 in figure 6.1 are oriented vertically and table 2 is oriented horizontally. We adapt our approach to work for both horizontally and vertically oriented tables. This is not difficult. We consider the origin table  $t$  and another table  $\bar{t}$  which is obtained from  $t$  considering its rows as columns and its columns as rows.

If  $t$  passes both the relational and header filter, table columns are considered as candidate attribute lines to extract attributes from. Similarly, if  $\bar{t}$  passes both the relational and header filter, the table rows are considered as candidate attribute lines. It can happen that both columns and rows are considered as candidate attribute lines. The latter are then passed to the attribute line filter.

### Attribute line filter

In addition to the relational and header filter, we defined the attribute line filter. Let  $a$  be the first cell of the line (row or column) and  $V$  be the rest of the cells of the line. We consider that a conform attribute line should contain an attribute name in the first cell and attribute values in the rest of the cells. Attribute values can correspond to one instance or multiple instances.

Typically, attribute names do not have much punctuation except of the colon and parenthesis. They rarely contain numbers. On the other hand, attribute values are usually in the same format (number, string, date) and their length is similar. Based on the above observations, we define the following features for the attribute line filter:

- presence of punctuation (except colons, brackets) in  $a$
- presence of numbers in  $a$
- $a$  is a stop word
- length (char) of  $a$ ;
- length (words) of  $a$
- average of the length (char) of values  $\mu$
- standard deviation of the length (char) of values  $\sigma$

- #values  $v \in V$  with  $|\text{len}(v) - \mu| > 2\sigma$
- #values  $v \in V$  with  $\sigma < |\text{len}(v) - \mu| \leq 2\sigma$
- #values  $v \in V$  with  $\sigma > |\text{len}(v) - \mu|$
- data conformity :  $\frac{\max_{T \in \text{int, string, date}} (\# \text{values of type}(T))}{|V|}$  (it measures at which extent the values have uniform type)

These features are then used to learn a rule-based classifier from a training set of human classified attribute lines described later in the experimental setup. Once candidate attributes are filtered, we rank the remaining set as explained in the following section.

### 6.2.2 Ranking attributes by relevance

It is not easy to tell whether an attribute is relevant for a given instance. There are many tables relevant to the instance where the instance is not even present in its cells. We propose combining different features to estimate a relevance score  $\phi(a, i)$  for a candidate attribute  $a$  and an instance  $i$ . These features include a score on the match of the instance on the table, document relevance and external evidence from DBPedia, Wikipedia and Web search.  $\phi(a, i)$  is a linear combination of these features. Relevance features are described below.

**Table match:** Tables from which attributes are extracted should be relevant for the instance. For some tables, this is easier to detect because the instance appears explicitly within the table cells. From an analysis on a small sample of random tables from the Web we derived two important observations. (i) There are many relevant tables that do not match the instance; (ii) however, there is a higher concentration of relevant attributes within tables that match the instance. This indicates that a table match feature will be helpful. We define it as follows:

Let  $a$  be an attribute extracted from a table  $T$  for an instance  $i$ . The match of an instance within a table cell  $T_{x,y}$  is measured with the cosine distance between the terms of the instance and the terms of the table cell. Let  $i$  and  $c$  be the vector representation of the instance and the table cell content. We have  $i = (w_{1,i}, w_{2,i}, \dots, w_{n,i})$  and  $c = (w_{1,c}, w_{2,c}, \dots, w_{n,c})$ . Each dimension corresponds to a separate term. If a term occurs in the instance, its value is 1 in  $i$ . If it occurs in the table cell content, its value is 1 in  $c$ . The match is computed with the cosine similarity among the vectors.

The table match score is computed as the maximal match within table cells:

$$\text{match}(i, T) = \max_{T_{x,y} \in T} \cos(i, T_{x,y}) \quad (6.3)$$

To improve the table match score, we rely on another important observation. We observed that an attribute name is unlikely to be present in

the same line (row or column) with the instance name, while the relevant attribute value is likely to appear in the same line with the instance. Indeed, the instance is more likely to be present in the first column or first row. Although its appearance in other cells is still evidence of relevance, it is unlikely that the attribute name and the instance will be met in the same line because in this case the instance is likely to be an attribute value. We can illustrate this with the table 2 in figure 6.1. The table matches the query “Paris” but the attribute “capital” cannot be relevant for this instance. To take into account for this observation, we will define the shadow area of a cell  $O$  as the set of cells in the same row and same column with  $O$ . There are some exceptions to this rule. We call headline cells, the ones that have are spanned ( $colspan > 1$ )<sup>3</sup> that cover the entire table width such as the ones in table 2 in figure 6.1. Headline cells usually act as titles that introduce parts of the table. We consider that the headline cells are not part of the shadow of another cell (see figure 6.2).

Figure 6.2: The shadow for a cell  $O$ , 3 cases

We define the match score of an attribute as the difference between the table match score and the shadow match score.

$$match(a, i, T) = match(i, T) - match(i, shadow(a)) \quad (6.4)$$

where

$$match(i, shadow(a)) = \max_{T_{x,y} \in shadow(a)} cos(i, T_{x,y})$$

**Document relevance** : If a document is relevant for the instance, the tables within the document are likely to be relevant for the instance. We should though take into account the document relevance. More precisely, let  $\#results$  be the number of retrieved results for an instance  $i$  and  $rank$  be the rank of a document  $d$  within this list. We compute:

$$drel(d, i) = \frac{\#results - rank}{\#results} \quad (6.5)$$

**Search hits**: Search hits is a feature that has already been used for attribute extraction [194, 141]. It corresponds to the number of search results returned by a Web search engine to a query “attribute of instance”

<sup>3</sup>The colspan is an HTML attribute that defines the number of columns a cell should span.

within double quotes (successive ordering of terms is obligatory, e.g. “capital of France”). As done in literature, we use the logarithm (base 10) of the search hits count. To normalize, we used the following observation. Few attributes score higher than 6 i.e  $\log_{10}(\text{search\_hits\_count}(a \text{ of } i)) > 6$ . All the attributes that score higher than 6 were given a score of 1, the other scores were normalized by 6. Doing so, we have all scores in the interval  $[0, 1]$ .

**DBPedia feature:** DBPedia represents a large ontology of information which partly relies on information extracted from Wikipedia [16]. Given an instance  $i$  and an attribute  $a$ , the DBPedia feature  $DBPedia(a, i)$  is equal to 1 if  $a$  is found as an attribute of  $i$  in DBPedia.

**Wikipedia feature:** Although information in DBPedia is much more uniform, there exist many attributes in Wikipedia *infobox* tables which are not present in DBPedia. *Infobox* tables are available for many Wikipedia pages. They contain lists of attributes for the page. Given an instance  $i$  and a candidate attribute  $a$ , we set the Wikipedia feature  $Wikipedia(a, i)$  to 1 if  $a$  can be found in the infobox of a page for  $i$  in Wikipedia.

**Other features:** It is difficult to find features which help rank attributes that are domain-independent and apply at large-scale. We excluded from the relevance features the *frequency feature*. This feature is meant to measure how often an attribute appears within tables of relevant documents. Intuitively, we can think that a relevant attribute will repeat more frequently than non-attributes or irrelevant attributes. We observed that candidate attributes that repeat the most are the ones that are used in ads or forms. Thus, candidate attributes such as “login”, “search”, “previous”, etc. were the ones that were favored by this feature. To tackle this issue, we developed a stop-word list for attributes, but even with this list, previous experiments did not show the interest of the frequency feature. This may be due to the fact that our list is still incomplete and need to be built using the whole web. We leave for future work a correct integration of this feature in the evaluation of  $\phi(a, i)$ .

**Combination of features** At last,  $\phi(a, i)$  is evaluated as a linear combination  $L$  of the relevance features:

$$\begin{aligned} \phi(a, i) = & L(\text{match}(a, i, T), \text{drel}(d, i), \\ & \log_{10}(\text{search\_hits\_count}(a \text{ of } i)), \\ & DBPedia(a, i), Wikipedia(a, i)) \end{aligned} \quad (6.6)$$

with  $T$  the table from which  $a$  was extracted and  $d$  the document containing  $T$ . We tried different combinations of these features to determine the contribution of each feature. The results are described later.

## 6.3 Experimental setup

In this section, we describe the experimental setup. We start with the general setup. We show how our approach was instantiated and then we follow with evaluation on filtering and attribute ranking.

For each instance of the dataset (200 instances), we retrieved top 50 search results using the Yahoo! BOSS API. These pages are used as a seed to extract tables and attributes. For each table, we apply the three filters sequentially, which are learned and evaluated as described in the next sections.

After filtering, we extract attribute lines and we rank them using the linear combination of features  $\phi(a, i)$ . More precisely, we use  $\phi(a, i)$  for instance attribute retrieval,  $\phi(a, I)$  for class attribute retrieval and  $\phi(a, I_+)$  for reinforced attribute retrieval.

Below, we will describe the evaluation and setup of filtering and attribute retrieval.

### 6.3.1 Filtering setup and evaluation

The three filters correspond to rule-based classifiers. They were trained using the previously mentioned features using the WEKA package [185]. We randomly selected a sample of 3000 tables (with more than 1 row and more than 1 column) which were judged by 3 assessors from our research institute. For each table assessors had to tell, if the table is relational. If “yes” they had to tell if the table is oriented vertically or horizontally and whether the table has a header.

Similarly, we choose a sample of 3000 random attribute lines from our dataset of tables. They are as well assessed from our assessors. For each attribute line, the assessor has to tell if it is a conform attribute line i.e. it contains an attribute name and attribute values.

We cross-validated the trained classifiers by splitting the human-judged dataset into five parts. Each time four parts are used training and the fifth for testing. We trained five different classifiers, rotating the testing set each time. Performance results are averaged from the resulting five tests.

### 6.3.2 Attribute retrieval evaluation

Attribute relevance was assessed by 5 volunteering participants from our institution. Each of the participants had to judge disjoint sets of attributes with respect to their relevance. An assessor had to consider an attribute relevant if at least its name was relevant. At a second time, assessors had to assess values from a sample of 5000 relevant attributes from the previously judged attributes. This is done for two reasons. First, we mentioned that attribute retrieval and attribute value retrieval can be considered as two

separate problems. Second, assessing attribute values can demand more time and expert knowledge. Though, we preferred assessing less attribute values. During evaluation, assessors could access the source page of the attribute or other sources (Web search, dictionary) to make their decision. Attributes were shown in alphabetical order to avoid any bias.

To evaluate the performance of our attribute retrieval approach we had to assess different runs of our approach and state of the art approaches. Initially, we used different combinations of relevance features. Then, we analyzed the impact of the filters and the retrieval performance across different search situations. To end, we had to assess the performance of state of the art techniques. Because there are far too many retrieved attributes and runs, we could not assess all of them. Instead, we assessed top 30 attributes from each run on a fixed dataset of queries. This is enough to estimate precision at rank with  $\text{rank} \leq 30$ .

We built a dataset of classes and instances to use as queries. This was done as it follows. First, we choose a set of classes and then 10 instances per class. To choose instances and classes, we used sampling to avoid biases. 5 participants had to write down 10 classes each. Classes could be broad (e.g. *Countries*) or specific (*French speaking countries*). We sampled 20 classes out of the 50. This is a reasonable amount as in state of the art approaches [174, 194, 28, 188, 135], the number of assessed classes varies from 2-25 classes.

Similarly for each selected class, we asked the 5 participants to write down 10 instances. Sampling and removing duplicates we obtained 10 instances per class. This is the list of classes: *rock bands, laptops, american universities, hotels, software, british army generals, chancellors of German, American films, IR papers, SLR cameras, novels, Nirvana songs, Nissan vehicles, programmable calculators, countries, drugs, companies, cities, painters, mobile phones*. The entire dataset can be found in the appendix.

## 6.4 Results

This section is about experimental results. Initially, we analyze the performance of the three filters. Then we analyze the performance of instance attribute retrieval which is the core element in our approach. This involves an analysis on the impact of filters and features for relevance ranking. Then, we analyze results by task namely attribute value retrieval, class attribute retrieval and reinforced attribute retrieval. Then, we show results on recall and comparison with state of the art.

### 6.4.1 Performance of filtering

Before analyzing the performance of the filters we analyzed all retrieved tables for all instances in our dataset. We found that only 16.9% of the

Horizontal			Vertical		
	prec.	recall		prec.	recall
yes	0.50	0.82	yes	0.38	0.81
no	0.98	0.94	no	0.98	0.95

Table 6.2: Precision and recall for the relational filter

Horizontal			Vertical		
	prec.	recall		prec.	recall
yes	0.96	0.95	yes	0.76	0.89
no	0.63	0.70	no	0.87	0.73

Table 6.3: Precision and recall for the header filter in relational tables

tables had more than one row and more than one column. Within the 3000 tables that were assessed only 23% were considered relational. We can thus estimate a concentration of 3.9% relational tables within the entire set of tables in the retrieved Web pages. Now, we will analyze the effect of the filters.

**Relational filter:** The relational filter is the first filter applied on the retrieved tables. This filter correspond to two different classifiers, one for relational tables that are oriented horizontally and another classifier for relational tables that are oriented vertically. The performance of these filters is shown in table 6.2.

We tuned classification for high recall over positives. The classifier of relational tables oriented horizontally retains 82% of the true positives and it filters out 94% of the true negatives. Similarly, the classifier of relational tables oriented vertically retains 81% of the true positives and it filters out 95% of the true negatives. After this filtering step, we will have about 45% of relational tables within our corpus out of an initial estimated concentration of about 3.9%.

**Header filter:** The header filter is applied on the output of the relational filter. In table 6.3, we show the performance of this filter. We observed that most of the horizontally oriented relational tables have headers. They usually have two to three columns and are easier to classify. The header classifier for relational tables oriented horizontally retains 95% of the true positives with a precision of 96%. Although the header classifier of relational tables oriented vertically is less performant, it retains 89% of the true positives with a precision of 76%. After this filtering step, about 87.5% of the relational tables will have headers and 71% of the tables without headers will be removed. Results on both filters are similar to the ones obtained by Cafarella et al. [37], although they are not directly comparable because they use another dataset.

**Attribute line filter:** The attribute line filter is applied on the output

of the header filter. As well as the other filters, the attribute line filter is tuned for recall over positives. The performance of this filter is shown in table 6.4. This filter retains 95% of the correct attribute lines, while it filters out about 55% of the incorrect attribute lines. It clearly helps in filtering out useless attribute lines at the cost of 5% of correct attribute lines.

	prec.	recall
yes	0.56	0.95
no	0.69	0.55

Table 6.4: Precision and recall for the attribute line filter in relational tables

In the next section, we will discuss on the effectiveness of the filters and the relevance features for the attribute retrieval task.

### 6.4.2 Impact of the relevance features on attribute retrieval

To determine the best ranking function  $\phi(a, i)$  we tried many different combinations of features. The number of runs we experimented is too big for us to enumerate all results. Indeed, if we suppose that the linear combination of features uses equal coefficients on all features and we just enumerate the possible combinations of features, we will have  $2^5 - 1 = 5 + 10 + 10 + 5 + 1 = 31$  different runs i.e. one features at a time (5 runs), two features at a time (10 runs), three features at a time (10 runs), 4 features at a time (5 runs) and all features at once (1 run). We will instead quickly resume the results.

First, we ranked using one feature at a time. The features that worked best in terms of precision are the DPBedia feature, Wikipedia feature and table match followed by search hits counts and document rank. After testing different runs, we observed that all the features can contribute to the performance of the ranking function. The best performance we obtained uses a simple linear combination with equal coefficients for all features. We acknowledge that light improvement is possible if we vary coefficients. For the sake of simplicity, all results shown here are obtained with the simple linear combination.

Performance of the best run is shown in figure 6.3. It shows precision at rank averaged over all instance queries. At rank 10, we have a precision of about 83% i.e. within the top 10 attributes about 8.3 are relevant. At rank 20 we have a precision of about 72%. Because retrieving attributes is presumably more difficult than retrieving documents, we can assume that these results are promising.

### 6.4.3 Impact of filters on attribute retrieval

In table 6.5, we show the performance of our approach with and without the filters. Symbol \* after the results of the first row indicates statistical



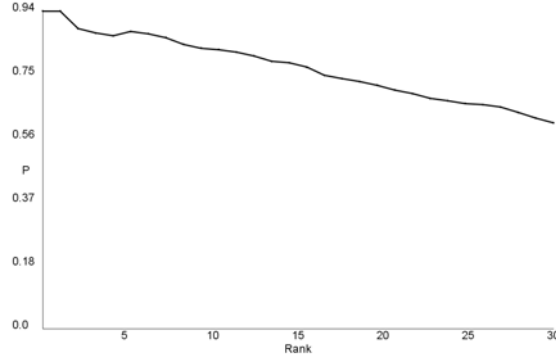


Figure 6.3: Precision at rank for instance attribute retrieval

significance using a paired t-test at  $p < 0.05$  (all filters against no filters).

We can see that all filters have a positive impact in the ranking as well as the relational and header filter. In fact, combining all three filters provides the best performance. However, results are encouraging with and without the filters. We can confirm one more time that ranking by relevance features works well.

Approach	p@1	p@10	p@20	p@30
All filters	0.94*	0.83*	0.72*	0.61*
Attribute line filter	0.90	0.82	0.70	0.61
Header and relat. filter	0.89	0.83	0.70	0.59
No filters	0.84	0.81	0.68	0.59

Table 6.5: The impact of filters

#### 6.4.4 Performance by search situation

In this section, we analyze the performance of different search situations. We start with attribute value retrieval following with class attribute retrieval and reinforced attribute retrieval.

##### Performance of attribute value retrieval

The above results concern only attribute names. We will introduce here some of our work on attribute values. Given a set of candidate attribute lines, we can rank attribute values in different ways. An easy way to rank is the following. First, we rank attribute lines with  $\phi(a, i)$ . Then, if the attribute line has two rows (or columns), we use the second cell as attribute value. If the instance name appears ortogonally with some cell (except the first) in the attribute line, we select this cell as the attribute value for the

attribute. Otherwise, we consider that the attribute value is the union of all cells (except the first). We call this attribute as multi-value. This simple method was shown to acquire correctly attribute values for 66% of relevant attributes in our dataset. These promising results will be completed by a complete analysis of attribute values retrieval in future work.

The rest of the results will concern relevance assessments on attribute names.

### Performance of class attribute retrieval

Figure 6.4 shows precision at rank for class attribute retrieval when we represent the class as a set of 10 instances. We retrieve attributes for each of the 10 instances of the class and we rank with the function  $\phi(a, I)$  defined in section 6.2 for the second problem. As expected, results are better than for instance attribute retrieval. We have precision of 0.95 at rank 10, a precision of 0.84 at rank 20 and a precision of 0.77 at rank 30. For the same problem, if we use 5 instances per class, we obtain precision 0.89 at rank 10, precision 0.76 at rank 20 and precision 0.66 at rank 30. In general, we noticed that class attribute retrieval performance improves with the increase of available instances.

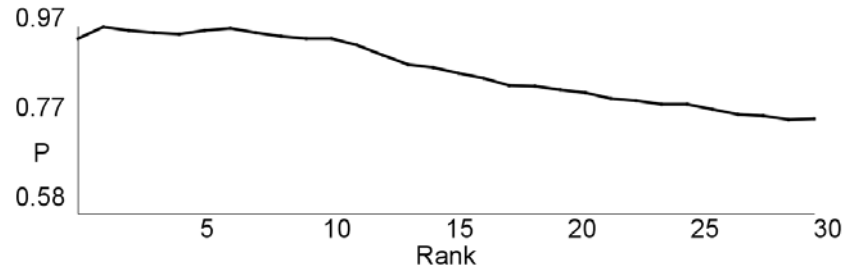


Figure 6.4: Precision at rank for class attribute retrieval

We also analyzed results class by class. Figure 6.5 shows precision at rank 30 by class. We can see that attribute retrieval can vary across classes. For some classes in our dataset, performance is lower such as for “drugs” and “programmable calculators”, while for others precision remains high such as for “british army generals”, “chancelors”, “countries”. The quality of attribute retrieval by class depends on the quality of attribute retrieval for every instance of the class. Results are relatively heterogeneous. Some instances are ambiguous. For some others, returned search results are slightly relevant. For some instances or class of instances, there is more tabular data than for other within search results. Furthermore, for some classes there exist more relevant attributes than for others.

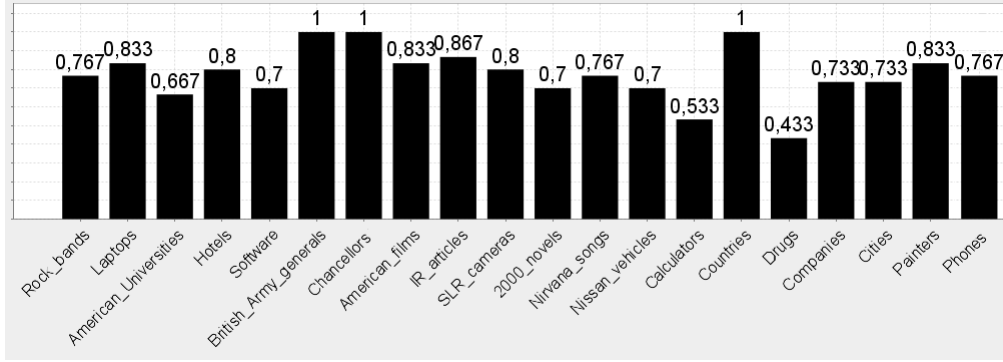


Figure 6.5: Precision at rank 30 by class

### Performance of reinforced attribute retrieval

To reinforce instance attribute retrieval, for each target instance we use the 9 other instances of the same class. Attributes are ranked combining both  $\phi(a, i)$  and  $\phi(a, I_+)$  as described earlier. The intuition is that the other instances can provide additional evidence of relevance for attributes.

In figure 6.6, we can see the impact of reinforcement on instance attribute retrieval. The lower curve corresponds to instance attribute retrieval without reinforcement, while the upper curve corresponds to the reinforced attribute retrieval. We can see that reinforcement improves significantly results<sup>6</sup>.

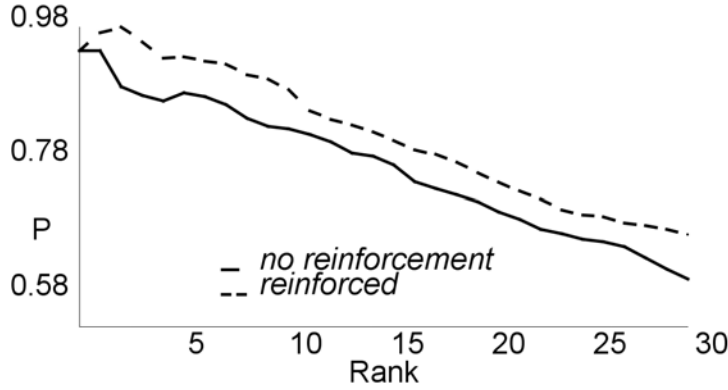


Figure 6.6: The impact of reinforcement

We can conclude that having multiple instances of the same class helps attribute retrieval for instance queries. We can explain this phenomena due to the fact that similar instances have similar attributes. Promoting common attributes has a positive impact on retrieval.

<sup>6</sup>The significance of the improvement was validated using a paired t-test with  $p < 0.05$  for P@10, P@20, P@30.

### 6.4.5 Estimating recall

To estimate in a reasonable time the recall of our method, we select 4 classes from our dataset namely “SLR cameras”, “countries”, “companies”, “Nissan vehicles” which are also met in Wikipedia and DBpedia. For all instances of these classes, our assessors evaluated all retrieved attributes (the ones that are not filtered out).

	cameras	countries	companies	vehicles	average
# retrieved attributes	689	1253	804	925	918
# relevant attributes	303	213	160	347	256

Table 6.6: Retrieved and relevant attributes by class and average

The number of retrieved and relevant attributes are shown in table 6.6 by class and average. We found an average of 918 distinct candidate attributes per class (SLR cameras 689, countries 1253, companies 804, vehicles 925). Among them, there are on average 256 relevant attributes per class (cameras 303, countries 213, companies 160, vehicles 347). This is a considerable amount of relevant attributes and it shows the potential of our method. It confirms that tables are a good source for attributes and that our filters keep a high concentration of relevant attributes.

We repeated a similar experiment for the same classes using Wikipedia and DBpedia. For each class, we randomly selected 10 instances which had to have a Wikipedia page or be present in DBpedia. We then extracted attributes which are either present in DBpedia or in Wikipedia infoboxes for each instance. We found an average of 25 distinct relevant attributes per class (cameras 8, countries 38, companies 37, vehicles 18). We observe that although Wikipedia and DBpedia are quality and huge sources of information, they have a much lower recall than our method.

### 6.4.6 Comparison with state of the art

We also compared our approach to the ones based on lexico-syntactic rules [28, 194, 174, 136] Lexico-syntactic rules are common for attribute extraction. We tested the lexico-syntactic extraction rules in our retrieval framework with our dataset. Concretely, we use the patterns “*A of I*” and “*I’s A*” as done in [136, 174]. We collect candidate attributes for 10 instances of all classes.

To compare we used the class attribute retrieval approach (problem 2). We use top 50 search results for all instances of our dataset as extraction seed for both techniques. The attributes extracted by the lexico-syntactic extraction method are ranked with the same scoring (excluding table match score which does not apply to lexico-syntactic rules).

Results are shown in table 6.7. Our method performs significantly better with 61% of relevant attributes at rank 30 against 33% for the lexico-syntactic rules. We recall that symbol \* after the results indicates statistical improvement using a paired t-test ( $p < 0.05$ ).

Approach	p@1	p@10	p@20	p@30
Our approach	0.94*	0.83*	0.72*	0.61*
Lexico-syntactic rules	0.46	0.48	0.43	0.33

Table 6.7: Comparison with lexico-syntactic scores

We also compared both approach in term of recall, by considering only the 4 classes used to estimate recall. Lexico-syntactic rules have a lower recall, too. For each class, lexico-syntactic rules identify 55 candidate attributes on average. Among these there are about 24 relevant attributes per class (against 256 for our approach).

We can conclude that lexico-syntactic rules work well for certain applications such as queries, but they do not work well for long documents, especially in terms of precision.

## 6.5 Conclusions

In this chapter, we present an approach for Web-scale attribute retrieval using relational HTML tables. Our approach integrates multiple features. Some of them are used to build filters to detect relational tables, the presence of headers and conform attribute lines. Other features are combined for relevance ranking.

All three filters prove their effectiveness. After applying the relational filter, we have about 45% of relational tables instead of an initial estimated concentration of about 3.9% of the initial corpus. The header filter if applied to relational tables, it retains 87.5% of the ones with headers, while about 71% of the tables without headers are removed. Similarly, the attribute line filter retains 95% of the correct attribute lines, while it filters out about 55% of the incorrect attribute lines. We can say that filters have a high recall over true positives and they filter out a huge amount of useless data.

The attribute retrieval approach is shown to perform well. We could observe that the approach has a high recall in terms of queries that can be answered and attributes that can be retrieved. Furthermore, ranking shows encouraging results in terms of precision. We could also show promising results when we experimented with three different uses namely attribute value retrieval, class attribute retrieval and reinforced retrieval.

Furthermore, we compare our approach to state of the art techniques. It is shown that it outperforms lexico-syntactic rules for the same purpose in terms of precision and recall. As well, we show that this approach has a high

coverage (recall) even when compared to quality sources such as DBPedia and Wikipedia.

The above proves that our attribute retrieval results are encouraging. In chapters 7 and 9, we will see how attributes can contribute to aggregate search results differently. For future work, we foresee the completion of the attribute retrieval framework with other recall-oriented or precision-oriented techniques. We believe that relations between classes, instances and attributes need to be further explored. As well, we will explore more on the issue of attribute value retrieval.

To conclude, this work proves that we can perform online attribute retrieval at Web-scale, while we can improve precision through ranking and validation techniques. As next step, we need to show how attribute retrieval will affect relational aggregated search. In the next chapter, we leave behind the attribute retrieval problems and we focus on result aggregation i.e. the construction of adequate answers composed of classes, instances and attributes.

## Chapter 7

# Result aggregation: Building useful tabular results

### 7.1 Introduction

In this chapter, we investigate on result aggregation for relational aggregated search. We have seen in the previous chapter that we can retrieve successfully attributes for instances and classes. Here, we focus on the assembly of instances and attributes for answering relational queries with a special focus on class queries.

Result aggregation is relatively simple for attribute and instance queries. For attribute queries, we can simply return candidate attribute values ranked by relevance. For instance queries, result aggregation is slightly more difficult. We need to return relevant attributes, but we also need to select the most important/representative attributes. The task becomes even more complex for class queries. We need to select and assemble important instances and attributes. Both the order of instances and attributes can affect the quality of the aggregated result. Because result aggregation for class queries is complex and it includes some of the issues met for other types of queries, we will focus in this chapter on class queries.

To illustrate the issues of result aggregation for class queries, we will list some examples using tables in figure 7.1. Let them be candidate results for the class query “French cities”. Each row corresponds to an instance and each column corresponds to an attribute. The first three tables contain obvious problems. Table **A** has some irrelevant attributes (e.g. email) and some useless attributes such as (e.g. yellow cards, golf players). Table **B** contains irrelevant cities (e.g. London) and some less representative cities such as Castres, Mulhouse, Roscoff. On the other hand, in table **C** we have issues for the attribute values. Most of the attribute values are missing. Table **D** is clearly more coherent and useful than the previous ones. It contains important (representative) class instances and important and relevant

	Country	Yellow cards	Golf players	Email	<b>A</b>
Paris	France	2	2000	paris@france.fr	
Marseille	France	3	1541		
Roscoff	France	NA	21		
Mulhouse	France	5	56		

	Population	Region	Area	Mayor	<b>B</b>
London	7,825,000	London	1,572.1 km <sup>2</sup>	Boris Johnson	
Roscoff	3,705		6.19 km <sup>2</sup>	Joseph Seité	
Castres	44,823	Midi-Pyrénées	457 km <sup>2</sup>	Pascal Bugis	
Mulhouse	110,514	Alsace	22.18 km <sup>2</sup>	Jean Rottner	

	Metro Population	Airport	Area	Metro Area	<b>C</b>
Paris	11,899,544	CDG, Paris Orly	2,723 km <sup>2</sup>	14,518.3 km <sup>2</sup>	
Roscoff	NA		6.19 km <sup>2</sup>		
Castres			98,17 km <sup>2</sup>		
Mulhouse			22.18 km <sup>2</sup>		

	Population	Region	Area	Mayor	<b>D</b>
Paris	2,211,297	Île de France	2,723 km <sup>2</sup>	Bertrand Delanoë	
Marseille	851,420	Provence	240 km <sup>2</sup>	Jean-Claude Gaudin	
Nantes	283,025	Pays de la Loire	65 km <sup>2</sup>	Jean-Marc Ayrault	
Toulouse	437,715	Midi-Pyrénées	118 km <sup>2</sup>	Pierre Cohen	

Figure 7.1: Examples of tabular result representation for class attribute queries

attributes with their values. The above examples are chosen to show that the quality of the aggregated search results depends on the quality of the instances, attributes and attribute values. Ideally, instances and attributes do not only have to be relevant but also important (representative) for the query (class in this case) and attribute values should be present and correct.

In this chapter, we define an approach that performs instance selection and attribute ranking for the construction of tabular results for class queries. The approach includes different weights on instances and attributes which indicate their utility to the query. We tested different combinations of the weights to measure and compare their impact on quality of the final results. In the next sections we will describe our approach following with the experimental setup and results.



## 7.2 Result aggregation approach

We have seen in the previous section that not all instances and attributes are equally important for a given query. In this section, we propose a result aggregation approach for relational aggregated search that can take into account for issues listed in the first section. To simplify the experimental setup we will suppose that candidate instances are relevant. This will enable us to focus more on novel issues that were not dealt in the previous chapter such as missing attribute values and the importance of attributes and instances.

Our approach is composed of two main steps *(i)* instance selection and *(ii)* attribute ranking. The first step provides the set of instances which will be used in the final result. For the given instances, we rank the attributes with respect to different weights trying to avoid missing values and promoting presumed important attributes. This setup allows us to test different rankings on attributes for different selections of instances.

To model our problem, we define different weights/scores on instances and attributes which will estimate the utility of these elements for the final result. These weights can be used to select and rank attributes and instances for result aggregation. We define weights on instances and attributes to fall in the interval  $[0, 1]$  and we assume that  $p_q(x) > p_q(y)$  indicates that  $x$  is more useful than  $y$  for the query  $q$ . We set  $p_q(x) = 0$  when  $x$  is irrelevant for the query  $q$ .

In our approach, we will test three different ways to select instances and three different ways to rank attributes. We will test the different combinations to see the impact of our weights in the quality of result aggregation. In the next sections, we propose the different weighting scores for instance selection and attribute ranking. Then, we show our experiments to test the utility of our weights for result aggregation.

### 7.2.1 Instance selection

We will define three ways to select instances which derive from simple hypotheses:

- **IH1:** *Instances with many attributes are likely to be important.* Indeed, if we can associate an instance with many attributes, it indicates that many authors use this instance. As well, these instances will likely contribute to fill a tabular result if we consider class queries i.e. more attributes means less empty cells.
- **IH2:** *Largely used and referenced instances are important.* This hypothesis relies on the idea that a popular instance (though largely used) is likely important.

The above statements are just hypotheses and they will have to be validated through experimentation. Using these hypotheses, we define three

ways to select instances. Each selection relies on a weight  $p_c(i)$  on a given instance  $i$  for a class  $c$ :

- **Maxy:** The first weight is derived from hypothesis **IH1**. Each instance  $i$  is scored with respect to the number of attributes it has normalized by the maximal number of attributes across all instances in the class.

$$p_c(i) = \frac{|attributes(i)|}{\max_{j \in I} |attributes(j)|} \quad (7.1)$$

where  $|attributes(i)|$  and  $|attributes(j)|$  are respectively the number of attributes of  $i$  and  $j$  and  $I$  is the set of instances of the class.

- **Linky:** We define a second weight relying on the second hypothesis (**IH2**) and state of the art approaches in TREC entity ranking track [20, 21]. The score we use is defined for instances that are found in Wikipedia. More concretely, the instance weight is defined using Wikipedia incoming links (*in*) to the instance Wikipedia page and outgoing links *out* from the instance Wikipedia page. This is similar to the work in [179]. Concretely, we have:

$$p_c(i) = \frac{in(i) + out(i)}{\max_{j \in I} (in(j) + out(j))} \quad (7.2)$$

where  $I$  is the set of instances of the class.

A similar score can be defined even for instances that are not present in Wikipedia. We need an estimation of the pages that refer to  $i$  and an estimation of important instances surrounding  $i$ . We will not define another score, because in our experimental setup, we will use instances that are found in Wikipedia.

- **Shorty:** We also define a third weight which decreases with the number of attributes an instance has. This opposite of the “maxy” weight. This weight is not supported by our hypotheses, but we will use it to test result aggregation with instances that have few attributes. For these instances the risk of having many empty cells in the final result is higher.

In the experimental setup, we will use the three selections of instances to influence the ranking of attributes. This is done to test if the notion of important instance relates to the notion of important attribute.

### 7.2.2 Attribute ranking

In this section, we define weights on attributes based on an intuitive initial formula:

$$p_c(a) = \sum_{i \in c} (p_i(a) \cdot p_c(i)) \quad (7.3)$$

with:

- $p_c(a)$  - the weight of the attribute  $a$  for the class  $c$
- $p_i(a)$  - the weight of the attribute  $a$  for the instance  $i$
- $p_c(i)$  - the weight of the instance for the class  $c$

We recall that for irrelevant attributes the weight is fixed to zero. When the attribute is relevant, its weight will be the sum of weights across multiple instances of the class.

To estimate weights on attributes, we use two different hypothesis:

- **AH1:** *An important attribute is likely to repeat frequently in a class (frequency.)* The intuition is that an attribute which is frequent across the instances of a class is likely to be important as most authors choose to use it.
- **AH2:** *An important attribute is present even when the instance has few attributes (participation) i.e. the attribute weight is disproportional to the number of attributes the instance has.* The intuition here is that if an instance is represented with just few attributes and if an attribute  $a$  is among them, then  $a$  is likely an important attribute.

The three weights we define will be named with respect to above hypotheses.

- **Frequency:** This weighting function is defined based on hypothesis **AH1**. Let *class frequency* ( $cf(a, c)$ ) be the frequency of attribute  $a$  in instances of class  $c$ . The frequency weight corresponds to  $cf(a, c)$  normalized on the maximum frequency i.e.  $|instances(c)|$ , the number of known instances for the class  $c$ . The frequency weight will be:

$$p_c(a) = \frac{cf(a, c)}{|instances(c)|} \quad (7.4)$$

This formula is equivalent to using equation 7.3 with  $p_c(i) = 1$  for all instances and  $p_a(i) = 1$  if the attribute  $a$  is met within the attributes of  $i$ .

- **Participation:** The “participation” weighting function is defined based on hypothesis **AH2**. We set  $p_i(a) = \frac{1}{|attributes(i)|}$  where  $|attributes(i)|$  indicates the number of attributes of  $i$ . The weight of an attribute  $a$  of  $i$  is therefore disproportional to the number of attributes  $i$  has. Replacing  $p_i(a)$  in equation 7.3 we obtain:

$$p_c(a) = \sum_{i \in c} \frac{1}{|attributes(i)|} \cdot p_c(i) \quad (7.5)$$

- **Frequency and participation:** In the third weighting function we combine the two above formulas (frequency and participation) in one. We define the weight as:

$$p_a(i) = \frac{cf(a, c)}{\sum_{a_j \in i} cf(a_j, c)} \quad (7.6)$$

We have that

$$p_c(a) = \sum_{i \in c} \frac{cf(a, c)}{\sum_{a_j \in i} cf(a_j, c)} \cdot p_c(i) \quad (7.7)$$

## 7.3 Experimental setup

We test our result aggregation approach on tabular results built for class queries. Our experimental setup is described in three steps. First, we present a dataset of queries (classes). Second, we present the different combinations of weights that were evaluated. Third, we present the human assessment deployment.

### 7.3.1 Dataset

To experiment our weighted result aggregation, we used 30 class queries selected randomly from DBPedia. The classes are listed in table 7.1. Only one constraint was imposed: classes had to have between 20 and 100 instances within DBPedia. For every class, we retrieved its instances within DBPedia and for every class instance we retrieve their attributes.

### 7.3.2 Tables

For every query we built tabular results containing 5 instances and 10 attributes per instance. Figure 7.2 shows how the result looks like (we do not show 10 attributes because they do not fit well in one snapshot).

We construct 9 different tables per query to investigate on the effectiveness of different attribute weights and for a possible correlation of attribute

Table 7.1: Class queries of the evaluation dataset

1. Colour	16. Seas
2. Albums produced by Mick Ronson	17. Manga of 1999
3. Archipelagos of Indonesia	18. English Journalists
4. Castles in Poland	19. Towns in Utah
5. European Union Member States	20. Burgess Shale Fossils
6. Defunct Agencies of the United States government	21. Countries bordering the Atlantic Ocean
7. Programmable calculators	22. Computing acronyms
8. Stoner rock musical groups	23. Tropical fruit
9. Academics of the University of Kent	24. Actors from Los Angeles/California
10. Turkish Riviera	25. G20 Nations
11. Radiocontrast agents	26. Adventure novels
12. Nissan vehicles	27. Ports and harbours of New Zealand
13. Australian World War I battalions	28. Poker companies
14. Number one debut singles	29. Markup languages
15. VIA Rail stations in Ontario	30. Chancellors

importance and instance importance. More precisely, we always show the same instances in all 9 tables for a given query, but we vary the attributes using the different ways to weigh that we defined. The instances correspond to the top 5 ones using the “maxy” selection, while attributes are ranked with respect to the 3 attribute weightings and 4 possible instance weightings.

The first attribute ranking is derived from the first weighting function namely “frequency”. The two other weights “participation” and “participation and frequency” demand for an estimation of  $p_c(i)$ . We estimate it through instance selection weights. We use one random selection of the instances and three selections using “maxy”, “shorty” and “linky” weights to rank top 5 instances.  $p_c(i)$  is set to 1 if the instance is within the selected instances and 0 otherwise. The combination of the 4 selections and the two attribute weighting functions makes 8 more possible ways to rank attributes.

Our goal is to compare the quality of the produced tables to derive the impact of each weighting function on the quality of results as well as the impact of instance weights to attribute weights.

### 7.3.3 Evaluation


Tables have been assessed from 4 volunteering participants from our institute. For each query, the assessor had to tell which table he/she prefers. Participants assessed disjoint sets of queries i.e. each assessor judge 7-8 queries and each query was judged by one assessor.

In addition, for every query we asked assessors to evaluate for relevance

	Family	Order	Class	Kingdom	Genus	Division
<b>Avocado</b>	Entité:Lauraceae	Entité:Laurales	Magnoliids Entité:Magnoliid	Entité:Plant	Persea Entité:Persea	
<b>Pineapple</b>	Entité:Bromelioideae Entité:Bromeliaceae	Entité:Poales Entité:Commelinids	Entité:Monocots	Entité:Plantae	Ananas Entité:Ananas	Entité:Angiosperms
<b>Eggplant</b>	Entité:Solanaceae	Entité:Asterids Entité:Solanales	Entité:Eudicots	Entité:Plantae	Solanum Entité:Solanum	Entité:Angiosperms
<b>Pomegranate</b>	Entité:Lythraceae		Entité:Rosidae Entité:Magnoliopsida	Entité:Plant	Entité:Punica Punica	Entité:Flowering_plant
<b>Banana</b>	Entité:Musaceae	Entité:Zingiberales Entité:Commelinids	Entité:Monocots	Entité:Plantae	Entité:Musa_%28genus%29	Entité:Angiosperms

Figure 7.2: Examples of tabular result to be assessed

all attributes that were used in the tables. This is done because there exist few attributes which are retrieved through DBPedia that are not relevant for users, which are mostly useful internally to DBPedia. We used these assessments to see if our weighting functions could filter out irrelevant attributes. The evaluation interface for attributes looks like in figure 7.3.




**Select the relevant attributes for the given query/class**

☐ elevation m   ☐ elevation (m)   ☐ mapsize   ☐ unit pref   ☐ latm   ☐ area total sq mi   ☐ blank info   ☐ area total  
☐ hasPhotoCollection   ☐ population total   ☐ map caption   ☐ blank name   ☐ subdivision name   ☐ area land sq  
☐ image skyline   ☐ latd   ☐ population density km   ☐ blank1 info   ☐ comment   ☐ reference   ☐ lats   ☐ longd  
☐ population as of   ☐ area water sq mi   ☐ population density (/sqkm)   ☐ Image   ☐ longs   ☐ utc offset DST  
☐ image caption   ☐ established date   ☐ named for   ☐ established title   ☐ homepage   ☐ imagesize   ☐ founder  
☐ lat s   ☐ lat d   ☐ wordnet\_type   ☐ long s   ☐ long m   ☐ lat NS   ☐ long EW   ☐ long d   ☐ lat m   ☐ Website  
☐ Aug Hi °F   ☐ Dec Precip inch   ☐ Jun Hi °F   ☐ Oct REC Hi °F   ☐ Feb REC Hi °F   ☐ Jan Lo °F   ☐ :

Figure 7.3: Example of the attribute assessment evaluation interface

## 7.4 Results

In table 7.2, we compare the impact of the attribute weights on the user preferences for the aggregated tables. The results show how often a table generated respectively with the ‘frequency’, ‘participation’, ‘frequency and participation’ weights are preferred by the user (in percentage). The ta-

bles generated with “frequency and participation” weighting function were more frequently preferred by assessors (60% of the cases). In second position, we find the “participation” weighting function. We can conclude that both “frequency” and “participation” weights are important, although “participation” is more important than “frequency”.

Table 7.2: Impact of attribute weights

	Frequency	Participation	Freq&Particip
Selection percentage	7.5%	32.5%	60.0%

In table 7.3, we compare the impact of the instance weights on the quality of tables. The results show how often a table with weights derived from the “random”, “shorty”, “maxy” or “linky” selection are preferred by the user (in percentage). The tables built ranking with the “linky” weight are the most frequently preferred by assessors (41.8% of the cases). This means that promoting presumably important instances produces better tables even better than promoting the attributes of the visualized table (“maxy”). As well, we observe that promoting the attributes of the instances with few attributes (“shorty” selection) has a negative impact. These instances are presumably less important or their impact increases the amount of empty cells.

Table 7.3: Impact of instance selection

	Random	Shorty	Maxy	Linky
Selection percentage	16.6%	8.3%	33.3%	41.8%

We will now show the impact of weighted ranking on the distribution of relevant attributes. This is a preliminary study, but results are interesting. In figure 7.4, we show precision at rank 10 ( $P@10$ ) for the attributes ranked using the three attribute weights in the presence of a random selection of instances. We can observe that there are surprisingly many irrelevant attributes on the top 10 attributes, but this is mostly due to the fact that many instances had less than 10 attributes. Though, for some classes it was impossible to have more than 5 relevant attributes at once. An interesting observation is that the weighting functions that work best are the same ones that produce preferable tables.

Table 7.4: Impact of attribute preferences on relevance  $P@10$ 

	Frequency	Participation	Freq&Particip
$P@10$	0.53	0.58	0.58

Figure 7.5 shows the impact of the instance weighting functions on the distribution of relevant attributes. We see that “linky” instance weight helps

to have more relevant attributes on top (P@10), followed by “maxy” and “shorty”. The only surprise is the “shorty” weight function which produces more relevant attributes than random selection of instances given that the tables with “shorty” ranking of instances are less preferred than tables with random instances. This can be easily explained through the fact that tables generated with the “shorty” selection can contain a lot of missing values. In general, we can confirm that the quality of the tables depends on the number of relevant attributes.

Table 7.5: Impact of instance selection on relevance P@10

	Random	Shorty	Maxy	Linky
P@10	0.58	0.583	0.595	0.595

## 7.5 Conclusions

We propose an approach for result aggregation in the context of relational aggregated search with a special focus on class queries. Our approach can take into account for different issues such as relevance and importance (an attribute/instance should be representative for the query), missing attribute values, etc. These issues are modeled through weights in different scoring functions. We promote attributes that have an important “participation” in the instances and a frequent appearance in the class. We also promote instances which are “popular” or instances that have many attributes. Results show that the weighting function can affect significantly the quality of tabular results for class queries.

For future work, we will experiment result aggregation with other variables. In particular, we would like to compare the impact of different relevance scores. In this chapter, we experimented with attributes and instances from DBPedia to avoid an excess in variables in the experimental setup.



## Chapter 8

# Lists of instances and set expansion

### 8.1 Introduction

In this chapter, we present an approach that enables to massively extract lists of instances. These lists were already shown useful to reinforce attribute retrieval. They have also been shown useful for querying by examples [184, 91], query suggestion [84], etc. Indeed, classes, instances and attributes are the main actors in relational aggregated search. The more we can identify and relate, the more information needs can be answered. The work we will present in this chapter is mainly about acquiring related instances to increase the extractable relational knowledge.

To identify instances of the same class, it is common to rely on the class i.e. we need to know the instance's class which will be associated with the other instances. For instance, to relate “Germany”, “Italy” and “France”, it is needed to pass through their class acquisition. Typically, this is done through lexico-syntactic rules [74] or existing knowledge bases such as DBPedia, Wikipedia, Wordnet [16, 75].

Binding the instance with a class name can be seen as a constraint, because this is not always possible and easy. One instance can belong to several classes (e.g Hotel California can match a song and a hotel). The named entity itself can be ambiguous (e.g Mr.Jones can refer to different persons, but also to a song). Moreover, it is difficult (see impossible) to agree on a common taxonomy of classes and we cannot enumerate a priori all possible classes.

Our approach follows a different paradigm with the goal of increasing the recall of relational data (both relations and extracts). We do not use any class acquisition technique, but we extract sets of instances in groups. We call these groups of anonymous class as *siblings' set* or *siblings' lists*. To illustrate, “*South Africa; Mexico; Uruguay; France*” is an example of

“siblings’ set”. These instances can be classified as “countries” but also as a “qualified teams for the FIFA 2010 World Championship”. Even if the class cannot be named univocally, the set itself incorporates most of the class’ semantics.

We use HTML lists as source for siblings’ set extraction. This is done for some reasons:

- HTML lists can contain siblings’ sets
- HTML lists are easy to parse
- There are many HTML lists in the Web

Nevertheless, there are some drawbacks with HTML lists. They are often used for their stylistic visualization rather than for a list semantics. We can often find them used for navigation menus, layout design, etc. Furthermore, the HTML language allows syntactic errors. Some tags are erroneous, some are not closed well. We keep into account for all these issues in this work.

The approach we propose is domain independent and applicable at large scale. Our contribution in this direction can be summarized with these points:

- We introduce the notion of siblings’ set and we situate it in the context of relational aggregated search
- We identify a rich source for extracting massively siblings’ sets within HTML lists.
- We analyze HTML lists and different features that might be discriminative for a domain-independent identification of siblings’ sets
- We implement a domain-independent classifier that allows to automatically extract siblings’ set from HTML lists
- We show one possible use of siblings’ sets namely set expansion (querying by example)

This chapter is structured as follows. First, we focus on the extraction of siblings’ lists. We present our approach, experimental setup following with results. We end this chapter with conclusions.

## 8.2 Mining for siblings’ list

Our approach is mainly about mining the HTML lists to learn extract siblings’ sets. We describe it through multiple steps. First, we describe the parsing process. Then, we list some “easy to identify” heuristics to filter out useless data. At this stage, we build a random dataset, which we use

to analyze the quality of non filtered HTML lists as well as to identify discriminative features. The dataset and the features are then used to train a classifier for automatic extraction of siblings' sets. Below, we detail these steps.

### 8.2.1 Parsing and filtering

**Parsing:** We consider only HTML lists i.e. lists within the respective HTML tags  $\langle OL \rangle$ ,  $\langle UL \rangle$  and  $\langle DL \rangle$ . The first tag represents ordered lists, while the others are unordered lists. Each list item is denoted by the  $\langle LI \rangle$  tag. The content of each item is within the opening and the corresponding closing tag as the content of each list is within the opening and the corresponding closing tag.

**Filtering:** Sometimes, it is not possible to parse correctly an HTML list. This is due to errors in the HTML structure both because of human or machine-generation coding. We could find a lot of tags which were not closed and tags which are miss-used. Our parsing is able to correct some of this errors. When not possible, it filters out the list.

At the same time, we filter out a large number of lists which were very probably not interesting for our study. This is done through heuristics. Some of these heuristics are easy to intuitive, for some others it was necessary to analyze data in advance. Concretely, we use the following heuristics to filter. We remove:

- lists with empty items
- lists with less than 3 items
- lists with items of less than 3 characters
- lists with the average length in characters of the items less than 4 characters
- lists with the longest element longer than 40 characters<sup>1</sup>
- lists with the average length in characters of the items greater than 32 characters<sup>2</sup>

After the parsing and filtering step, we analyzed the result of this process in a sample of 1000 documents. We found about 3.4 lists per page and about 5.7 items per list. We also notice that the length of the lists varies a lot, where the longest list has 466 items. We observe that in average 85% of the HTML lists do not pass the filtering process. This is initial evidence that many of the HTML lists are used for layout-design and other purposes.

---

<sup>1</sup>chosen by observations

<sup>2</sup>chosen by observations

### 8.2.2 Mining and extracting

To analyze the potential of HTML lists for massive extraction of siblings' sets, we use a labelled dataset of HTML lists randomly taken from a corpus of documents. These lists will be helpful to estimate the distribution of siblings' sets. This will also be used to mine for useful features that can help extracting siblings' lists. Below, list the features that we use in our study followed by a list of potentially useful features (future work):

- **Links (anchors):** Many Web Designers often use lists to show navigation options (menus), to propose related links or simply for advertisement links. This is very frequent and one can expect to find fewer lists of instances within these cases. It is common to expect links (anchors) within the item of such lists. This is why we studied the influence of links in the quality of lists.
- **Frequencies:** In analogy with term frequency in a document ( $tf$ ) or the document frequency ( $df$ ), we define the item frequency ( $if$ ) for lists as in how many lists an item is found in the lists of the collection. We study if such a frequency has a role in the quality of lists. For instance menu items can be very frequent such as "*Home*".
- **Ordered versus unordered:** It is interesting to study whether there are more lists of instances within ordered or unordered lists.
- **Item count:** Another feature we studied is the influence of the number of items. Are short lists more probable to contain instances of the same type or longer ones behave better?
- **Other features to consider:** The above analysis helps to analyze the distribution of siblings' lists in HTML list with respect to specific features of HTML lists. These features might not be enough for an optimal classification of siblings' list. Here we list some other features that can be used. They can be linguistic, domain specific, collection specific and so on.

For each of the terms in a list item, it is possible to generate specific features using collection statistics or external sources such as WordNet. A promising idea is the use of statistical functions [176, 44] such as the Pointwise Mutual Information (PMI) which estimate the relationship among two or more items based on the co-occurrences of their terms.

It is also interesting to consider other domain independent features such as the presence of punctuation, the presence of verbs, the position of the list in the table, etc. In this work we limit to the specific features of HTML lists. We want to show their utility and their limits.

After analyzing the distribution of siblings' list with respect to each feature, we devise an automatic classifier to extract siblings' lists. All selected features are domain-independent and common for HTML lists. They are listed in table 8.1. They include list length, the list type (ordered or not), the item length in terms of characters, item frequency and the presence of links. The presence of links is passed as two different features, one indicating the presence of at least one link in the list items, another indicating if all items are linked. The values are transformed into real numbers and normalized when necessary.

Table 8.1: The features used for classification

Feature	Description
Type	Is the list ordered?
Length	The number of items of the list
NoLinks	There is no anchors at all in the list
AllLinks	All items in the list have anchors (links)
MaxIF	The item with the maximal item frequency
AvgIF	The average item frequency
MinIF	The item with the minimal item frequency
Longest	The maximal length in characters of an item
AvgLgth	The average length in character of items

We rely on a SVM classifier, because it works well in the presence of few features. Here, kernel functions can map the input dimensions into a higher dimensional space which allows to classify data which are not easy to separate in their hyper-space.

We use the open source LIBSVM library [42], which provides support vector classification, (C-SVC, nu-SVC), regression (epsilon-SVR, nu-SVR) and distribution estimation (one-class SVM). These three types of classification can be used with 4 types of kernels namely the radial based function, the sigmoid function, linear combination and polynomial combination.

### 8.3 Experimental setup

We used the QUAERO<sup>3</sup> corpus which contains pages extracted from the Web. It counts about 1 million and a half Web pages, which are in French language. The collection is a subset of the index of Exalead Search Engine<sup>4</sup>. Even if our goal is to generalize our experimental results, we are aware that

---

<sup>3</sup>QUAERO is a project among French and German public and private research organizations aiming to promote research and industrial innovation on technologies for automatic analysis and classification of multimedia and multilingual documents, <http://www.quaero.org>

<sup>4</sup><http://www.exalead.com>

this number is not large enough with respect to the size of the Web and that there might not be the same distribution of lists in French language as for other languages.

To evaluate the quality of the HTML lists in terms of presence of siblings' list we use human assessors. We are aware that there exist various automatic classifiers such as the Stanford NER<sup>5</sup> [59], ABNER<sup>6</sup> [157], etc. for instances, but we opted for human classification because automatic classifiers do not work well for all classes and instances (we aim a domain independent approach).

We relied on 5 human participants which assessed 2000 lists randomly selected. Assessors are all from our institute in the domains of Information Retrieval and Information Systems. It is easier to provide such a task to experts of these domains, because they are familiar with the concepts of named entity, relation and class. Consequently, less training time is needed for assessors and better assessments can be obtained. In fact, we are not interested in knowing whether an average human can identify instances, more than identifying them correctly.

Each assessor was shown the same instructions, which were first tested for comprehensibility i.e. other assessors were asked whether the stated instructions and definitions were clear and unambiguous. The final guidelines included the definition of named entity, examples of instances and examples of lists of instances of the same type.

The task of the assessor consists mainly in checking if a given list is a siblings' set of instances or not. The question was simplified to the assessors. The answer is "yes" when:

- all list items are instances
- the instances belong to the same class

Otherwise the answer is "no".

The assessor was told that the named entity should be the entire list item and not simply present in the item. For lists longer than 5, one wrong element was marked as acceptable (e.g. France, Germany, Italy, Spain, Albania, Others).

The interface the assessor had to interact with shows the list and the "yes/no" question on the top of a Web page, followed by the source page below. This is clearly useful as no one knows all about all instances. In difficult cases, which were very rare, assessors could use search engines.

---

<sup>5</sup><http://nlp.stanford.edu/ner/index.shtml>

<sup>6</sup><http://pages.cs.wisc.edu/~bsettles/abner/>

## 8.4 Experimental results

In this section, we provide our analysis on the labelled dataset, the impact of each feature on the distribution of siblings' sets. Then, we focus on the performance of our classifier.

### 8.4.1 Distribution of siblings' sets

From 2000 assessed lists only 165 of them were judged as lists of instances of the same type. This is only 8.25% of them. If we consider that we filtered 85% of the lists and that there are 3.4 lists per page, it is possible to estimate for our dataset that there are at least 3.99 siblings' lists each hundred documents. This is a small proportion, but it is a strong indicator that the Web with its size is a rich mine for extracting siblings' lists.

In fact, with the above estimation, if we consider 1 billion documents, we can expect 40 million siblings' lists and if we consider an estimation of the number of indexed pages in 2009 which is 22.3 billions<sup>7</sup>, we can expect to extract about 892,000,000 siblings' lists which is huge and would be probably the largest collection of useful lists ever collected.

To validate this estimation, it might be necessary to repeat the same experiment with a larger dataset, at least one order of magnitude. As well, it will be interesting to check whether language and cultural differences affect the use of lists and their quality.

### 8.4.2 Mining by feature

Below we consider some important features which can help to detect subsets with larger percentage of siblings' lists. This is useful to understand which features are discriminative and for future work to design adequate classifiers.

**Links (anchors):** When it comes to links, we distinguish 3 cases. The first one is lists where all items have links ("all links" case). The second case is lists where at least one item has no links ("some links" case). In the third case, we consider only lists which do not have any link at all ("no links" case).

The result are shown in table 8.2. The first column shows the case. The second column shows how many lists from our filtered sample share fall in each class, followed by a percentage relative to the size of the sample. The percentages are useful to understand how frequent lists of each type are.

The results show that the lists where all items have links are less probable to be siblings' sets. Among lists with at least one item which is not linked there are 17.5% which are siblings' lists, while if there are no links at all the results are even better (22.5%).

---

<sup>7</sup>Taken from <http://www.worldwidewebsite.com/> on October 2009

Table 8.2: Quality and number of lists for link related features

Set	Siblings' lists %	Number of lists
All lists	8.2%	59756
All items linked	7.5%	53441 (85.6%)
Some links	17.5%	6345 (10.6%)
No links	22.5%	1750 (2.9%)

Although among lists with all items with links only 7.5% are siblings' lists, we cannot neglect this subset. They represent about 85.6% of the lists of our list dataset, which makes this set a very important seed to find siblings' lists.

**Item frequencies:** Table 8.3 shows the items which are the most frequent in lists. As one can see, these are mostly items that help navigation, but not instances. In fact, most of them appear mostly with links.

We observed that if a list has an item which repeats very frequently with links, the list is less probable to be qualitative. For instance, among lists where there exist an item that has a frequency (*if*) above 160 times<sup>8</sup>, the percentage of good lists is only 3.2%. If we consider the average of the item frequency over all list items, we observe that among lists with an average item frequency above 40 times<sup>9</sup>, only 0.7% are qualitative.

There are surprisingly a lot of lists with a maximal item frequency above 160 (24273 lists about 40.6%) and average item frequency above 40 (27131 lists about 45.4%). The above statistics can be very useful to filter out potentially useless lists leaving a subcollection with almost two times better quality.

On the other hand, focusing on lists for which all items appear only once, we found 40% of siblings' lists, which is much above the dataset's average.

Table 8.3: The most frequent items in the lists of our dataset

Name	English translation	Frequency
Accueil	Home	3887
Forum	Forum	1819
Contact	Contact	1506
Recherche	Search	1448
Musique	Music	1291
Inscription	Subscription	1246
Accueil forums	Forums home	980

**Ordered versus unordered lists:** It is interesting to study whether there are more lists of instances within ordered or unordered lists. The first

---

<sup>8</sup>chosen by observations

<sup>9</sup>chosen by observations



interesting observation is that there are much more unordered lists in the web than ordered ones. In facts, in our dataset we observed that ordered lists represented only 0.4% of the entire set.

Therefore, the quality of the entire set of assessed lists is almost determined from the quality of the unordered lists. We had to demand assessors to judge specifically ordered lists as the number of assessed ordered lists was very low.

The result is surprising. About 60% of the ordered lists are siblings' lists. Although, there is only 277 of them within the 59756 lists, the result is encouraging as the quality is much higher than for unordered lists. This can be explained by the fact that ordered lists are closer to the traditional meaning of list and are less frequently used for layout design.

**Item count:** Another feature we studied is the influence of the number of items. Are short lists more probable to contain instances of the same type or longer ones behave better? The graphic in figure 8.1 shows the obtained results per value.

The results are shown for the range of 3 to 16 items. The later values are not shown in the graphic as there are not many assessed lists for the remaining values.

We distinguish 4 ranges that behave differently. The range 3-6 has an average quality of 6.8%. The range 7-9 has an average quality of 12.5%. The range 10-15 has an average quality of 4%, while if there are more than 15 items the average quality is 15%.

The results tell that there are twice as many quality lists in the range 7-9 with respect to the range 3-6. It also tells that there are more quality lists for long lists (more than 15 items).

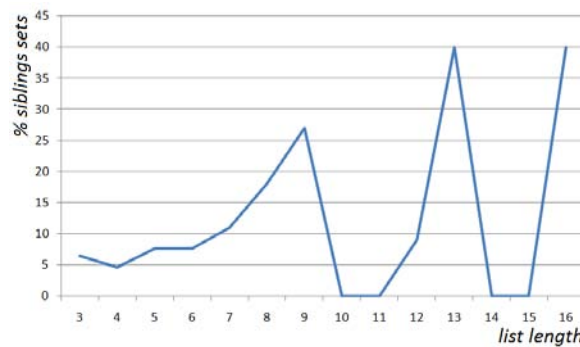


Figure 8.1: % of siblings' lists with respect to the list length

### 8.4.3 Extracting siblings' lists through classification

In this section we propose a binary classification for HTML lists, which aims to detect if a list is a siblings' set or not.

We tested all three types of SVM classifiers (vector classification, regression and distribution estimation) with the 4 kernels (radial based function, sigmoid function, linear combination and polynomial combination). In each case, different combinations of features were also input to see if there are feature combinations which are more discriminative than others.

To validate results a 3 fold cross-validation is used. This consists in dividing the dataset in 3 subsets, where 2 are used for training and 1 for testing. The choice of the test set is then altered three times and the classifiers performance is better estimated by an average of the three turns. As measures, we use precision and recall for both the positive class (siblings' set) and negative set (other lists).

It is impossible to show all the obtained results for all combinations of features and configurations. If the classifier answers always "yes" it obtains a 100% recall for the siblings' sets, but a low precision. In table 8.4, we show 4 interesting cases to compare. We take a *always-yes* answering classifier and an *always-no* answering classifier. We put them aside with two well performing classifiers, one that performs particularly well in precision and another that obtains good recall at the cost of precision loss.

Table 8.4: Precision and recall of the classification

	Positive		Negative	
Classifier	Prec.	Recall	Prec.	Recall
Always Yes	8.25%	100%	-	0%
Always No	-	0%	91.75%	100%
C-SVC <sup>15</sup>	57.1%	18.1%	91.5%	98.4%
ONE-CLASS <sup>16</sup>	14.0%	44.7%	92.2%	71.0%

Our results vary in terms of precision and recall. It is possible to have a 100% recall, but the precision remains low. Reasonable results for precision and recall are obtained with the third (C-SVC) and the forth classifier (ONE-CLASS).

The third classifier is used with a classical configuration and it uses all the listed features. It obtains encouraging results if we consider that there is a relatively low concentration of siblings' lists. The forth classifier performs better in terms of recall. This can be due to the fact, that this type of classifier works well for detecting outliers.

The results remain encouraging as we address a domain independent massive extraction of lists of instances of the same type. Training sets of instances with their classes as well as other state of the art techniques in named entity extraction can be integrated in such a system to improve

<sup>15</sup>Features considered: all

<sup>16</sup>Features considered: presence of links, the type of list (ordered or not) and the item frequency

precision and recall while remaining in the context of a massive extraction. We also believe that the introduction of new features and especially the PMI measure will significantly improve results. These results show the benefits and limits of HTML list specific features.

## 8.5 Conclusions

In this chapter we propose an investigation for a massive and domain-independent extraction of lists of instances of the same class (siblings' sets). Instead of using class acquisition, we automatically classify list of candidate instances into siblings' sets. This approach is tested with HTML lists from the Web.

Our experimental results are interesting. We show that about 8.25% of the HTML lists in our randomly selected dataset are siblings' sets. If our estimation is validated and can be generalized to the Web, using an estimation of the size of the Web, we can estimate to have more 892 million lists of instances of the same type, which corresponds to billions of instances.

The identification of such a huge collection of instances of the same type is only one part of the contribution. Further analysis on our assessed lists is used to understand which features are more discriminative for siblings' sets.

The observations are used to build automatic binary classification to detect lists of instances. The performance of the classifier varies with respect to the features used and the chosen configuration. We obtain 57.1% of precision and 18.1% of recall if we bias precision. We obtain 14% of precision and 44.7% recall if we bias recall. These results can be further improved with the integration of new features such as Web-scale term correlation statistics. We also believe that adapting this work for online retrieval will improve performance. The encouraging result is that these huge collection of instances exists and it is precious for future exploitation.



## Chapter 9

# Completing relational aggregated search and prototypes

### 9.1 Introduction

In this chapter, we present different prototypes we could build applying our research on relational aggregated search. We also introduce some new investigation of ours on different sub-problems which can complete relational aggregated search. This work is particularly interesting because we highlight different components of RAS systems. We can see this as a journey towards the composition (construction) of complete relational aggregated search systems. Our overall contributions is summarized below:

- **Querying:** We list different ways to query a relational aggregated search system (section 9.2). This includes structured queries and natural language queries.
- **More retrieval:** We extend the experimented retrieval approaches with some light retrieval techniques for relational aggregated search (section 9.3). In particular, we describe some of our research on attribute value retrieval and passage retrieval.
- **Result aggregation:** We show different ways to put together information in the context of relational aggregated search (section 9.4).
- **Flexible relational aggregated search:** We show a simple prototype where we can deal with different types of queries at the same type. Each query type is answered with the corresponding preferred result aggregation (section 9.4.4).

Class	Instances
countries	<div>1. France</div> <div>2. Italy</div> <div>3. Germany</div> <div>4. Spain</div> <div>5. USA</div> <div>6. Australia</div> <div>7. South Africa</div> <div>8. Albania</div> <div>9. Tunisia</div> <div>10.</div>

[revoir](#) [Help](#)

Please insert a class name and at least 4 instances e.g. class countries , instances France, Italy, Spain, Albania

---

Prev queries: [SIG IRIT](#) [Actors](#) [music albums](#)  
[Adjectives](#) [Animals](#) [Country](#)  
[sites](#) [formula 1 drivers](#) [american universities](#)

Figure 9.1: The class query issued as through multiple text boxes

We present our prototypes starting from their components. First, we focus on the querying language. Second, we show briefly how to retrieve other types of content such as attribute values, passages and images. After showing these elements, we describe the prototypes one by one and we illustrate with examples. Each of these prototypes has its own result aggregation.

## 9.2 Querying

As we mentioned earlier in our framework, we distinguish three types of queries namely class queries, attribute queries and instance queries. We define different ways to write each of these queries. We end with a unified approach to write all these queries in quasi-natural language.

- **Class queries:** Class queries can be input in natural language such as “countries” or “European countries”, but we will use multiple instances to represent class queries, because we did not deal in our prototypes with instance retrieval. Though, we try two different ways to input class queries. The first one consists of dedicated text boxes which allow up to 10 instances (see figure 9.1, we input a name for the class just for logging purposes). The second option is through a set of instances separated by commas (see figure 9.2). This allows illimited number of instances, although it is less intuitive for some users.
- **Attribute queries:** We use two different ways to input attribute queries. The first option is through two different text boxes as shown in figure 9.3. This way, there is no confusion between the attribute name and the instance. The second option is through natural language

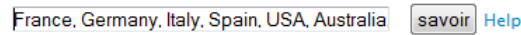


Figure 9.2: The class query issued as instances separated by commas

e.g. “president of France”, “GDP of UK”, “author of Iliad” (see figure 9.4). The main issue with this approach is that sometime the instance name itself can contain the word “of”. For example, “wind of change” corresponds to a song and “wind” is not an attribute of “change”. To avoid this kind of issues, we allow the use of brackets to delimitate instances when the query can be interpreted ambiguously. Though, we can write “album of (wind of change)”.



Figure 9.3: The attribute query issued through two text boxes




Figure 9.4: The attribute query entered through natural language

- **Instance queries:** Instance queries are the easiest, because they do not need to be decomposed into smaller subqueries. The only problem we meet with instance queries comes when the query can be of whatever type. In the next section, we propose a more flexible querying approach which allows to input the three types of query (class query, instance query and attribute query) through one unique text box.
- **Untyped relational queries:** What happens if we do not know the type of query that is being issued? The ideal solution is to determine their type at query time, but this is not trivial. Alternatively, we propose a simple approach that allows inputting the three types of relational queries. If the query contains the word “of” outside of brackets, then we consider the query as an attribute query. If the query contains items separated by commas, then we consider the query as a class query. Otherwise the query is an instance query. Ambiguous entities (the ones that contain the word “of”) have to be surrounded by brackets. We can also query for an attribute across multiple instances if we separate instances by commas after the term “of”. The below examples will illustrate<sup>1</sup>:

<sup>1</sup>The quotes are not part of the query. They are used to separate queries from each

- attribute queries: “capital of Argentina”, “population of Los Angeles”, “album of (Wind of change)”, “GDP of Italy, France, Germany”
- instance queries: “France”, “Los Angeles”, “(Wind of change)”
- class queries: “Italy, France, Germany”, “Always somewhere, (Wind of change), Holiday”

### 9.3 More retrieval: attribute values, passages and images

In this section, we show some investigation we did to integrate into the relational framework content of other types except of classes, instances and attributes. In particular, we focus on attribute value retrieval, passage retrieval and image retrieval which are built on top of Yahoo! BOSS API search results. They are described below.

#### 9.3.1 Attribute value retrieval

We present two techniques to retrieve attribute values for an attribute query. In the previous chapters, we have seen a simple technique to extract attribute values during attribute retrieval. This technique has a 66% precision on attribute values. But sometimes, the attribute we are interested in is not retrieved by instance attribute retrieval. Here, we propose two simple techniques to retrieve the attribute value of a given attribute and instance. The first method simply extends attribute retrieval on tables. The second technique targets pure text.

**Attribute value retrieval from tables:** To increase chances of finding an attribute value, we use as retrieval seed a set of documents retrieved on queries derived from the attribute name and instance. Let  $a$  be the attribute name and  $i$  instance. We issue as queries “ $a$ ” “ $i$ ” (including quotes). Quotes are used to guarantee a precise match with the instance and attribute name. As well, we issue the same query without quotes, because sometimes the same instance can be found with different names. The retrieved seed will be composed of results on both queries. Then, we apply the same extraction and scoring as described previously in attribute retrieval.

**Attribute value retrieval from text:** Our second attribute value retrieval method relies on two hypothesis. First, the attribute value (if present) can be found close in text with the attribute name. Second, the document should be relevant for the instance. Though, we use as seed a set of top documents retrieved on the queries  $i$ ,  $a$   $i$ , “ $a$ ” “ $i$ ” and “ $a$  of  $i$ ” where  $a$  and  $i$  are respectively the targeted attribute name and instance.

---

other.



As candidate values, we consider terms or consecutive terms near occurrence of the attribute name  $a$ . These candidates are scored on different features including: distance from  $a$ , frequency of each term, presence of stop words, presence of punctuation, etc. Although this method is not yet mature, it shows encouraging results.

### 9.3.2 Passage retrieval

Passage retrieval [86, 150, 182, 109] is not easy in general, which can be explained from the lack of success in Web search. Nevertheless, passage retrieval can be successful in conjunction with other information retrieval techniques such as attribute retrieval. In this section, we propose a simple approach to extract and rank passages. The approach is far from state of the art, but it integrates useful thoughts from relational aggregated search.

In particular, we interest in *instance-passage* relations, which will be identified through retrieval and mining. We make some analogy between attributes and passages. They can both have a name (title for passages), but passages content is usually longer than attribute values. Similarly to attributes, we assume that common passage titles will repeat across similar instances. We also assume that passages from Wikipedia are more qualitative than others.

Our approach combines different heuristics defined on the passage length, the query match on the title and on the body of the passage. Passages are ranked through a simple score calculated by our heuristics  $\beta(p)$ . If multiple instances are available, we average over the rest of the instances to provide reinforcement to ranking as done with attributes.

Even if this is ongoing work, our main goal here is to show that passages fit well in the relational aggregated search framework.

### 9.3.3 Image retrieval

We integrate results from image retrieval through the Yahoo! BOSS API. Images are frequently useful for named entities [170]. Even when they are not relevant, they are not very harmful when they are not the primary target of search.

## 9.4 Result aggregation and prototypes

In this section, we will list the various prototypes we developed to play with relational aggregated search. They allow us to illustrate the utility of RAS. In particular, they include different ways to aggregate results. They prove that relational aggregated search is possible and results are promising.

The prototypes can be accessed in the address <http://www.irit.fr:8080/force/><sup>2</sup>.

To start with in figure 9.5, we show the main menu we use to access all developed prototypes. All prototypes are named with French verbs that end in “*oir*”. There is no specific reason behind the name choice, except it sounds good. The “*valoir*”, “*pouvoir*” and “*revoir*” prototypes correspond respectively to retrieval through attribute queries, instance queries and class queries. The prototype “*savoir*” integrate all of these queries into one flexible relational aggregated search system. All prototypes are tuned with a light configuration to allow a reasonable answer delay (seconds to maximum 1-2 minutes). We will next briefly describe the first 4 prototypes.



Figure 9.5: The main menu used to access all developed prototypes

#### 9.4.1 Prototype “*valoir*”

“*Valoir*” is an prototype dedicated to attribute queries. Queries are input through two text boxes, one for the attribute name and one for the instance. The candidate attribute values are retrieved from tables and pure text using the methods described earlier in section 9.3.1. We show to the user the top three attributes retrieved from each method. The user can ask for more results if he wants.

A result example is given in figure 9.6. The query is “president of France”. In this case, attribute retrieval from tables (top 3 results) seems working better. We prefer keeping ambiguous results, because we want to clearly identify the contribution of each technique.

In the image, we can also see the search results that were used to retrieve the attribute values. The returned values can come from one or multiple different documents. In future work we want to introduce links within the results to easily access the source result of each extract. Source results

<sup>2</sup>The prototypes will soon move from Yahoo! BOSS API to Microsoft’s Bing API

<b>Results for "president of France ":</b>	
	sarkozy nicolas sarkozy nicolas nicolas sarkozy first ladies eduard shevardnadze ...
Sources:	
<a href="http://en.wikipedia.org/wiki/President_of_France">http://en.wikipedia.org/wiki/President_of_France</a> <a href="http://en.wikipedia.org/wiki/Nicolas_Sarkozy">http://en.wikipedia.org/wiki/Nicolas_Sarkozy</a> <a href="http://en.academic.ru/dic.nsf/enwiki/15278">http://en.academic.ru/dic.nsf/enwiki/15278</a> <a href="http://www.wordiq.com/definition/President_of_France">http://www.wordiq.com/definition/President_of_France</a> <a href="http://www.trueknowledge.com/q/who_is_the_president_of_france_2010">http://www.trueknowledge.com/q/who_is_the_president_of_france_2010</a> <a href="http://www.worldlingo.com/ma/enwiki/en/President_of_France">http://www.worldlingo.com/ma/enwiki/en/President_of_France</a> <a href="http://www.bonjourlafrance.com/france-facts/president-of-france.htm">http://www.bonjourlafrance.com/france-facts/president-of-france.htm</a> <a href="http://www.wikinfo.org/index.php/President_of_France">http://www.wikinfo.org/index.php/President_of_France</a> <a href="http://wn.com/President_of_France">http://wn.com/President_of_France</a> <a href="http://topics.europeanvoice.com/topic/about/Presidents+of+France">http://topics.europeanvoice.com/topic/about/Presidents+of+France</a>	

Figure 9.6: The “valoir” result on the query “president of France”

are also shown in other prototypes, but we will omit them in the next examples.

### 9.4.2 Prototype “*pouvoir*”

“*Pouvoir*” is an prototype dedicated to instance queries. Queries do not need any special parsing or multiple text boxes. They are simple input in natural language. In this prototype, we combine three types of content in the answer namely attributes, images and passages. On top we place top retrieved attributes. The user can click on the attribute names to view their values. Another option is to show values right away. The second line contains 3 images retrieved with Yahoo! BOSS. Then, we list in two columns passages.

Figures 9.7 and 9.8 show results of this prototype respectively for the queries “Ireland” and “Pink Panther”. We can see that for these queries all types of content contribute with different relevant aspects. The final result looks indeed as a document, although it was generated automatically.

### 9.4.3 Prototype “*revoir*”

“*Revoir*” is a prototype dedicated to class queries. Queries are input through as a set of instances in a dedicated querying form that can take up to 10 instances (see figure 9.1). The result is a table of instances, attributes and their values. Here, we use the attribute retrieval method described in chapter



Figure 9.7: The “pouvoir” result on the query “Ireland”

6, but with a light configuration to avoid long answering delay.

In figure 9.9, we can see the result on the query “music albums”. Here, we issued the query through 7 instances which are well-known albums. We use the word album after each instance to avoid ambiguity. We can see that retrieved attribute names are relevant, while their values are somehow messy. This is the case because at this stage we are experimenting with attribute values. This interface is not at its final stage. We present multiple candidate values to see if ranking of values works well.

#### 9.4.4 Prototype “savoir”

“Savoir” is the most complete of our prototypes. It accepts all three types of relational queries and it proposes a tailored solution for each type of query. Queries are input in quasi-natural language (see section 9.2 for untyped relational query). This querying convention allows to determine the query type through simple parsing. For each query type, we trigger the adequate solution.

To have results presented in a uniform format, we use a table which has instance names as columns and attribute names as rows. If the query is an attribute query, the user will be presented the retrieved values within a



Figure 9.8: The “pouvoir” result on the query “Pink Panther”

	Image	genre	label	length	recorded
American Life album		pop , electronica , rock , dan pop , electronica , rock , dan	maverick , warner bros.   maverick   warner maverick , warner bros.	49:39 49:39    [ 4:58 ; 4:24 ; 4:09 ; 3:38 ; 4:39 ; 4:49 ; 4:54 ; 3:50 ; 4:33 ; 4:38 ; 5:05 ] [ 4:58 ; 4:24 ; 4:09 ; 3:38 ; ; ]	june 2002
Hstory album		pop , r&b , new jack swing , h   r&b , pop , rock , dance , urb	epic epic	71:39 (disc 1) 77:06 (disc 2)   71:39	september 1994
Nevermind album		grunge   grunge rock: alternative	dgc   dgc geffen	42:38 42:38    [ 5:01 ; 4:15 ; 3:39 ; 3:04 ; 4:17 ; 2:57 ; 2:22 ; 3:44 ; 2:37 ; 3:32 ; 3:16 ; 3:51 ]	may?june 1991
invincible album		r&b , pop , dance-pop[]   r&b , pop rock , electronic   r&b , pop , dance-p	epic   motown , epic , legacy   epic ardent recordsforefront record   epic	77:08   77:08 52:38   77:08    [ 6:26 ; 5:09 ; 4:46 ; 5:29 ; 4:49 ; 5:39 ; 4:40 ; 3:18 ; 4:24 ; 4:33 ; 5:05 ; 4:24 ; 5: ; ; ]	october 1997
reload album		rock   other   heavy metal   rock hard rock , heavy metal   rock	gut/v2   gut/v2 elektra   gut/v2   elektra	62:36   75 min 56 sec   62:36 76:07   62:36    [ 3:37 ; 2:58 ; 3:25 ; 3:58 ; 3:26 ; 3:39 ; 3:31 ; 3:48 ; 3:18 ; ; ]	1998-1999
the score album		hip hop , soul , reggae hip hop , soul , reggae	ruffhouse , columbia   warner ruffhouse , columbia	73:32 73:32	june-november 1995
the wall album		progressive rock , rock opera   r&b , disco , funk , dance pop progress	harvest records / emi records	81:09   42:16 81:09    [ 6:04 ; 3:40 ; 5:14 ; 4:39 ; 4:05 ; 3:05 ; 3:37 ; 4:29 ; 3:48 ; 3:41 ]	january?november 1979

Figure 9.9: The “revoir” result on the query “music albums” issued through class instances

simple table headed by the instance name and the attribute name. Instance queries are answered with attributes and their values and few images. Class queries are answered similarly, but their table has multiple columns, one per

France	
<i>president:</i>	sarkozy nicolas sarkozy nicolas nicolas sarkozy first ladies eduard shevardnadze ...

Figure 9.10: The “savoir” result on the query “president of France”

Mac Book Pro	
<i>developer:</i>	apple inc. ...
<i>predecessor:</i>	powerbook g4 ...
<i>release date:</i>	january 10, 2006 (original release) february 24, 2 ...
<i>type:</i>	laptop ...
<i>operating system:</i>	mac os x 10.4.4 and later ...
<i>website:</i>	apple ? macbook pro ...
<i>days since update:</i>	1 (avg = 229) ...
<i>last release:</i>	february 24, 2011 ...
<i>recommendation:</i>	buy - product recently updated ...
<i>current:</i>	airport · displays · isight · magic mouse · magic ...
<i>mac imac, mac pro, mac mini:</i>	apple: i, ii, iii; lisamacintosh: compact; ii; se, ...
<i>discontinued:</i>	apple: i, ii, iii; lisamacintosh: compact; ii; se, ...

Figure 9.11: The “savoir” result on the query “Mac Book Pro”

	Nokia e72	Samsung Galaxy	iPhone
<i>connectivity:</i>	wlan wi-fi 802.11 b/g, integrated & assisted gps ...	usb 2.0, bluetooth 2.1, wi-fi b/g, gps ...	...
<i>cpu:</i>	600 mhz arm 11 processor ...	arm11 528 mhz + dsp 256 mhz ...	...
<i>dimensions:</i>	114 x 59.5 x 10.1 mm ...	115 x 56 x 11.9 mm ...	...
<i>display:</i>	320x240 px (0.1 megapixels), 2.36 in, up to 16.7 m ...	320 x 480 px , 3.2 in, amoled , touchscreen ...	...
<i>form factor:</i>	bar ...	candybar ...	...
<i>memory:</i>	250 mb internal user storage rom: 512 mb sdram: 12 ...	128 mb ram ...	...
<i>op. system:</i>	s60 3rd edition feature pack 2 ui on symbian os ...	android v1.6 (donut)originally android 1.5upgrada ...	...
<i>rear camera:</i>	5 megapixel (2592 x 1944 pixels) with autofocus a ...	5 megapixel with auto focus; 720p hd video(12mbps ...	...
<i>weight:</i>	128 g ...	114g ...	...
<i>manufacturer:</i>	nokia ...	samsung ...	...
<i>predecessor:</i>	nokia e71 ...	samsung galaxy i-7500 ...	...
<i>comp networks:</i>	gsm 800 / 900 / 1800 / 1900 mhz tri band utms / hs ...	dual band cdma2000 /ev-do rev. a 800 and 1,900 m ...	...

Figure 9.12: The “savoir” result on the query “mobile phones” issued as instances separated by commas

instance.

In figure 9.10, we see how results look like for attribute queries. Figure 9.11 shows results for the instance query “Mac Book Pro”, while figure 9.12 show results for the class query “mobile phones” issued through three instances (we reduced the result on the third instance, because it did not fit into the page.). We can see that result presentation is relatively uniform and readable.

We believe that these results are encouraging.

## 9.5 Conclusions

In this chapter, we have shown 4 prototypes of relational aggregated search. The first three are dedicated to one type of query respectively to attribute queries, instance queries and class queries. The forth prototype allows all three types of queries and answers each type of query with the appropriate designated solution. We can see this chapter as a prototype oriented trip within relational aggregated search. During this trip, we highlight issues and possible solutions. The combination of all solutions is a relational aggregated search system with encouraging performance.





## Part IV

### Part 4: Cross-vertical aggregated search: Interest and evaluation



## Chapter 10

# Interest and evaluation of cross-vertical aggregated search

### 10.1 Introduction

Cross-vertical aggregated search can be seen as a special case of federated search. The interest (advantages, novel issues) of this research direction remains to be explored. The work presented in this chapter targets at the same time interest and evaluation of cross-vertical aggregated search. We consider different ways to collect relevance assessments in this context trying to determine the advantages of cross-vertical aggregated search. The goals are multiple. On one side we want to study why and how multiple and diverse sources can be useful. On the other side, we want to identify the best ways to capture this utility.

We found that there are two types of relevance assessments that are used in literature for *cvAS*. In [14, 104, 108], human judges (assessors) are given a query and they have to assign to it to one or more vertical intents if they find so. *A query has a vertical intent if there exists a vertical search that is likely to answer the query.* In this kind of setup, assessors do not know the real need behind the queries and they are not shown any concrete results from search engines. We say that we have “*relevance by intent assessments*” on “*short text queries*”.

In [166, 168], Sushmita et al. investigate some of the advantages of cross-vertical aggregated search interfaces. They show that cross-vertical AS increases the quantity and diversity of relevant results accessed by users. Here, human assessors are shown results from each source being used and queries are associated with a description of the information need. We say that we have *relevance by content assessments* on *queries with a fixed information need* (fixed need queries). Until now, there exist no studies that

relate or compare the two types of relevances (by intent and by content). As well, the impact of the fixed information need with respect to the short query (free to interpret) has not been studied in the context of cross-vertical aggregated search.

Our goal in this work is to reconsider the evaluation of the interest of AS, by exploiting both relevance by intent and by content, and by using queries with or without fixed information need. Our research questions include:

- What is a relevant source?
- How realistic are relevance by intent assessments and relevance by content assessments?
- Depending on the evaluation setup, which is the distribution of relevant sources?
- Which is the contribution of vertical searches to traditional Web search?
- Why can two or more sources be complementary at the same time? Are different sources complementary to each other?
- How should we setup evaluation of cross-vertical aggregated search?

Some of these questions have already been examined in literature (see sections 5.6 and 5.2 for details). Our aim is to revisit interest and evaluation of cross-vertical aggregated search by exploiting two types of relevance (by intent and content) and two types of queries (fixed need and short query). For this purpose we conducted a study with many human participants. We examined four search situations where we vary the type of relevance and the type of query.

This chapter is structured as follows. Sections 10.2 and 10.3 introduce our study setup and its results. In section 10.4 we provide discussion about the results and we give some thoughts about the evaluation of *cvAS*.

## 10.2 Experimental setup

We built a study which involves relevance assessments collected through human participants. The goal is to determine the advantages of cross-vertical aggregated search as well as the evaluation issues. We aim to investigate on the notion of source relevance and the ways to capture it.

Our study is composed of 4 tasks which involve real participants evaluating relevance across 9 different sources. We consider two types of source relevance (by intent or content) and two types queries (with or without a fixed information). The two types of relevance and the two types of queries are defined below.

### 10.2.1 Definition of source relevance

Intuitively, a relevant source is the one that can provide useful information for an information need. In this context, source relevance is different from search result relevance. A source might be relevant for a query even if no relevant results exist for this query. We can define source relevance with one of the following statements:

- **Definition 1 - Relevance by intent:** a source is relevant for a query if it makes sense to issue this query to this source (i.e. there can be a user intent to use this source). The source does not even have to be concrete. For instance, the abstract source “image search” is relevant for the query “Eiffel tower photos”. Relevance assessment does not depend on the quality of any search engine.

- **Definition 2 - Relevance by content:** a source is relevant for a query if it contains relevant results for the query. The source has to be an existing search engine and relevance assessment depends on the quality of the source and the availability of relevant results for the query.

### 10.2.2 Queries

We randomly sampled 100 queries from the Million Query Track in TREC 2007 [40], which contains itself queries extracted from search engine logs. This is done for the following reasons. First, choosing manually queries or writing down ourselves queries can bias results. Second, we could not find a collection of query logs free to use, which would be the ideal solution. The choice of the Million Query Track can be a bias as queries are likely to have textual intent<sup>1</sup>. However, this is not a major issue as our goal is to compare evaluation dimensions rather than the real distribution of query intents. Selected queries are listed in the appendix.

In the following, we distinguish among **short-text query** and **query with fixed need**. A short-text query corresponds to a query as it is provided by TREC (1 to 3 words). We associated to each query a detailed description (a TREC-like textual description) corresponding to what we considered as relevant for this information need. This is what we call query with fixed need.

### 10.2.3 Sources and participants

We consider 9 sources: 8 vertical searches and Web search. The sources had to be diverse with minimal intersection. Most of them are common well known vertical search engines. They were chosen for two main reasons. First, they have already been used in state of the art approaches [14, 166, 167]. Second, we could find a free API for the source. The list of sources

---

<sup>1</sup>There is at least one match in the GOV2 collection.

is the following: *Web search, Video search, Image search, News search, Maps (geographic search), Wikipedia search, Product search, Answer search, Definitions.*

For evaluating relevance by content, we used the following real sources:

- Yahoo! BOSS to get results for Web, image, news, answers and Wikipedia,
- API-s offered by Google Maps, Tuveo, E-Bay and Bing for geographic search, videos, products and definitions respectively.

The study was conducted with 33 participants belonging to our research institute with 14 master students, 14 PhD students, 6 lecturers and 1 engineer. There were 13 women and 22 men.

Each participant had to fill a questionnaire before the test to collect data on participants and their experience with search. Another questionnaire was to be filled after the test where the participant is asked questions about the task and his test.

#### 10.2.4 Evaluation interface

When evaluating relevance by content, we use a simple way to show search results which corresponds to an unblended search approach. This is done for the following reasons. First, we do not propose any aggregation algorithm that ranks results of different sources. Second, we want participants to access results from all sources without favoring any source. Results are shown in 9 panels (see figure 10.1) forming a 3x3 square. Each panel is labeled by its source and contains only results from that source (vertical search or Web search). To avoid biases, sources are shown in a random order for each query.

Participants might be skeptical about results coming from one source or they might expect results in one source and not in another accordingly to the issued query. To avoid this kind of bias results are shown in a homogeneous manner and the participants are instructed to view results from all sources. The study is conceived to produce no bias among sources except of the results quality itself.

The number of results per source to be shown is chosen based on the visualization space and the comprehensibility of the results. For geographic search, only one result is shown in the map. The panel for images is filled with 9 images. For the rest of the sources 3 results are shown for each. It was possible to show 9 videos, but videos are not as self explanatory as images. Showing a title with some description is more informative.

Figure 10.1 presents an example of the interface shown to participants for the query “Eiffel Tower”.

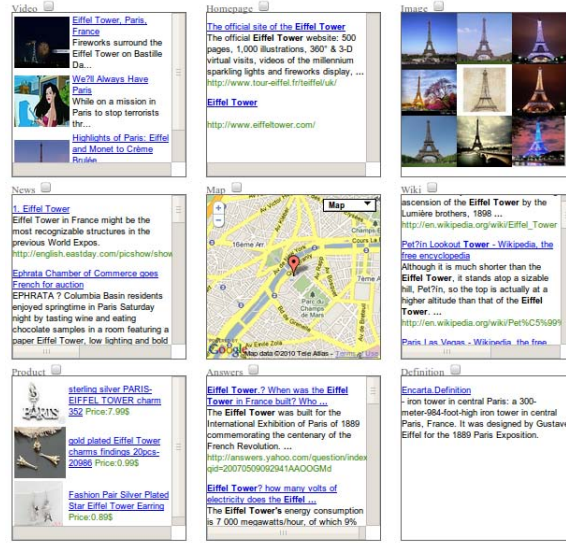


Figure 10.1: The interface for evaluating relevance by content with the results for the query “Eiffel tower”

### 10.2.5 Tasks description

We evaluated 4 search configurations named here tasks, each one corresponding to one type of relevance and one type of queries.

- **Task 1:** short-text query, relevance by intent
- **Task 2:** short-text query, relevance by content
- **Task 3:** fixed need, relevance by intent
- **Task 4:** fixed need, relevance by content

Those tasks are detailed below.

#### Task 1

This task is similar to evaluation in [14, 104, 108]. In this task, the participants have to decide for each short-text query which sources are likely to be relevant (relevance by intent). In other terms, they have to choose in which sources they expect to have useful results for that query. They are free to have all possible interpretations for the query. If they do not get any interpretation for the query they have to say so. Table 10.1 shows an example for the query “London Tower”. Different sources such as Web, Wikipedia and images can be relevant.

Query: <i>London Tower</i>				
Video	Web	Image	News	Map
No	Yes	Yes	No	Yes
Wiki	Product	Answers	Definition	
Yes	No	No	No	

Table 10.1: Example of relevant and irrelevant sources for the query “London Tower”

### Task 2

In this task, each short-text query is submitted to the 9 vertical searches. Participants are shown results from all sources (if some results exist for the considered source) in the interface presented in section 10.2.4. This task is not meant to evaluate the way of presenting results (through the interface), but the utility of each source in the context of limited visualization space as well as the utility of the whole combination of sources and results.

A source is considered relevant if it returns at least one relevant result, whatever the interpretation of the query. The participant is indicated to view results from all sources, even if he has an interpretation in his mind and even if he does not expect relevant results for a given source.

### Task 3

Here the participants are given a fixed need query (with the description of the information need that could have triggered the query). Using both the query and the information need, they have to choose the sources they guess to be relevant (relevance by intent).

### Task 4

In this last task, the participants are also given a description of the information need that could have triggered the query as well as they are shown results from each source. This task is closer to traditional IR evaluation. As for Task 2, a source is considered relevant if it introduces at least one relevant result (relevance by content).

The participants are also asked to tell which is the most useful source for the information need. If they can choose one, they have to tell if they find this source enough for the information need. This allows us to investigate deeper the advantages of AS.

#### 10.2.6 Task deployment

All tasks except the second involve 10 participants with 30 queries each. The second task involved 30 participants with 10 queries each. We limited



the number of queries, because this task takes longer time than the other tasks. A participant who participated in all tasks would evaluate 100 queries. Tasks were designed to have every query assessed by exactly 3 participants, therefore we have 300 queries assessed by task.

### 10.3 Results

In this section we first discuss the results concerning the interest of cross-vertical aggregated search and then we compare evaluation results by considering different types of relevance and queries.

Before giving these results, we list in the following section the notations we used.

#### 10.3.1 Notation

In the figures, graphs and tables of this section, we will use the following abbreviation for the sources: *video* for *video search*, *web* for *web search*, *wiki* for *Wikipedia*, *map* for *geographic search*, *prod* for *product search*, *image* for *image search*, *def* for *definitions*, *qa* for *answer search*. We will use  $U1$ ,  $U2$ ,  $U3$  and  $U4$  to refer respectively to task 1, task 2, task 3 and task 4 of our study. Furthermore, we need some definitions. Let  $q_i^u$  be the  $i^{th}$  query assessed by participant  $u$ . Let  $Q$  be the set of all  $q_i^u$  in a task,  $|Q| = 300$  for each task.

Finally, let  $Q(s_j)$  be the set of  $q_i^u$  for which the source  $s_j$  is considered relevant.

#### 10.3.2 Interest of cross-vertical aggregated search

The first question we try to answer is whether it is worth aggregating results from different vertical search engines in the context of Web search. Before getting to the heart of this question, we first define the *average relevance of a source  $s$*  as the proportion of queries for which  $s$  was relevant. Relevance might be assessed by intent or content depending on the task being considered. We will denote it as  $R(s)$  and it corresponds to:

$$R(s) = \frac{|Q(s)|}{|Q|} \quad (10.1)$$

Results here will be discussed according to the goals mentioned at the beginning of section 10.2.

Table 10.2 lists the average relevance of sources with respect to the different tasks. The same data is also shown in figure 10.2 to ease comparison between different sources. We can see that the Web source is the most relevant and definitions source is the less relevant whatever the considered relevance or query type. If we order sources by average relevance, we notice

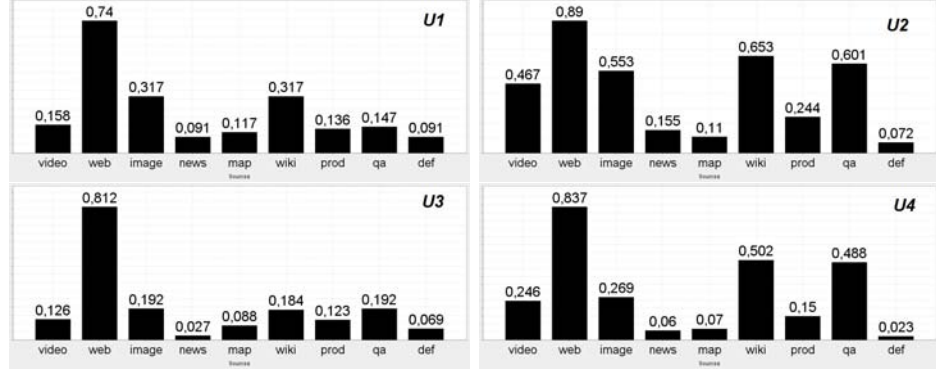


Figure 10.2: Relevance of sources based on  $U1$  (top left),  $U2$ (top right),  $U3$ (bottom left) and  $U4$ (bottom right)

	Video	Web	Image	News	Map	Wiki	Prod	QA	Def	Total
$U1$	0.15	0.74	0.31	0.09	0.11	0.31	0.13	0.14	0.09	2.11
$U2$	0.46	0.89	0.55	0.15	0.11	0.65	0.24	0.60	0.07	3.74
$U3$	0.12	0.81	0.19	0.03	0.08	0.18	0.12	0.19	0.06	1.81
$U4$	0.24	0.83	0.26	0.06	0.07	0.50	0.15	0.48	0.02	2.64

Table 10.2: Comparative table of the average relevance of sources with respect to tasks

that not all tasks produce the same ranking. However all 4 tasks confirm that **relevance is sparse across all sources**.

We also compare the relevance of vertical searches with Web search with respect to the number of queries that can be answered respectively by Web search and the union of all verticals. Let  $A$  be the set of all sources,  $web$  be the Web search,  $V$  be the set of all verticals ( $V=A-web$ ). We will say that  $R(A)$  represents the proportion of queries where there is at least one relevant source,  $R(V)$  represents the proportion of queries where there is at least one relevant vertical. We also denote as  $R(web, \bar{V})$  the proportion of queries where the Web search is relevant alone (no relevant vertical) and  $R(V, \bar{web})$  the proportion of queries where there is at least one relevant vertical but Web search is considered irrelevant. Results are shown in table 10.3. The last column corresponds to an average over all tasks.

We notice (in row  $R(A)$  of the table) that about 98% of queries have at least one relevant source. Web search is relevant for about 82% of queries, while 78% of queries have at least one relevant vertical. Furthermore, Web search is relevant alone (no relevant vertical) for only about 19% of queries. The interesting result comes from the last row of the table where for about 16% of the queries have at least one relevant vertical without having Web search as relevant. We can draw the following conclusions.

Study	U1	U2	U3	U4	avg
$R(A)$	0.99	1.00	0.99	0.97	0.98
$R(web)$	0.74	0.89	0.81	0.83	0.82
$R(V)$	0.75	0.92	0.66	0.79	0.78
$R(web, V)$	0.24	0.08	0.31	0.17	0.19
$R(V, web)$	0.25	0.11	0.18	0.13	0.16

Table 10.3: Duration and agreement by task

**Vertical sources can answer many queries. Mostly they are relevant at the same time with Web search, but they can also present relevant results when Web search fails to.**

Let us now examine if many sources can be relevant at the same time and if sources are complementary with each other. Concretely, we want to analyze how often multiple sources are considered relevant and when this is the case we want to see if multiple relevant sources provide same information (same results) or they return different results for a given query.

Before discussing deeply this question, we first counted the number of queries that have more than one relevant source. We found 155 queries for task 1, 251, 134 and 215 for tasks 2, 3 and 4 respectively. There are also some queries with 4, 5, 6 even 7 relevant sources. The average number of relevant sources per query and per task is in fact listed in the last column of table 10.2. It is respectively 2.11, 3.74, 1.81 and 2.64 for the tasks  $U1$ ,  $U2$ ,  $U3$  and  $U4$ . We can therefore conclude that **many queries match more than one relevant source**.

### 10.3.3 Complementary sources?

An important part of our analysis was spent to understand the reasons of having more than one relevant source for a given query. We identified three major reasons: *(i)* The query is ambiguous. *(ii)* The query is broad, though the information need is composed of multiple aspects (e.g. visit France can demand for maps, images, Web, etc. ). *(iii)* Two different sources can return the same information. Although we chose a diverse set of sources, there are cases when two different sources present the same information. The most frequent is Web search which can intersect with all other sources.

We investigated further the last two reasons (*(ii)* and *(iii)*). We examined whether the sources return mostly same results (same information) for a query or whether they tend to return complementary information that help to better answer the query (in that case we call these sources complementary). For this purpose, participants were asked in task 4 to select the most relevant source (called primary source) and to tell if this source was sufficient to satisfy the information need or if the other sources help (if they are complementary).

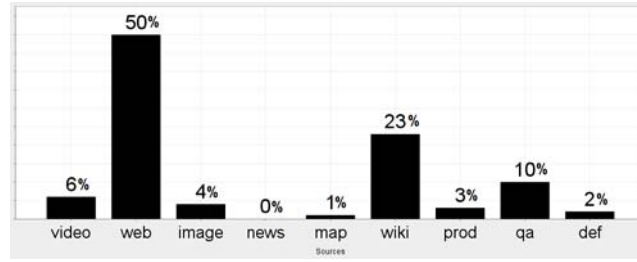


Figure 10.3: Distribution of the primary sources

Among 300 cases (10 participants x 30 queries), in 272 cases a most useful source was selected. Figure 10.3 shows the distribution of primary sources. One notices that the Web is the most useful source half of the times, for the other half it is the other sources that present the most useful information. For this half, the vertical search is more directly relevant to the information need than the Web search.

The most useful source alone was considered sufficient for the information need 106 times (39%) and it is insufficient 166 times (61%). If the source is not enough it is necessary to provide additional complementary information. Apparently, this is needed quite often.

The participant was also asked to tell if he/she finds additional useful information in other sources for his/her information need. For the 106 cases where the most useful source was considered sufficient alone, an average of 2.13 other sources were selected as useful. We can say that the participant found the primary enough for his information need, but at his/her surprise he could meet more relevant information in the other sources. For the 166 cases where the most useful source was considered insufficient alone, an average of 2.74 other sources were selected as useful. We can expect these sources to provide complementary information which is missing in the primary source. **Results show implicitly that cross-vertical aggregated search is useful because often one source alone is not enough to answer an information need, while multiple sources can complement each other for a complete answer.**

Let's consider a relevant source  $s$  as complementary to a relevant source  $p$ , when  $p$  is a primary source and not sufficient alone to satisfy the information need. This is not an exact definition, but we will use it to analyze the distribution of complementary sources. More precisely, we investigated whether some sources are more likely to be complementary to a given source  $p$ . Figures 10.4 and 10.5 show the distribution of complementary sources respectively for the Web source and Wikipedia source. We see that the distribution of complementary sources varies with respect to the primary source.

For instance, the source which is more often complementary to the Web

source is the answers source followed by Wikipedia, video search and image search. On the other side, the source which is more often complementary to Wikipedia is the Web search source followed by image search and then answers source. **We can therefore say that some sources are more likely to complete a given source.**

This is just a short analysis of possible relations between primary and complementary sources. Further investigation is needed in this direction.

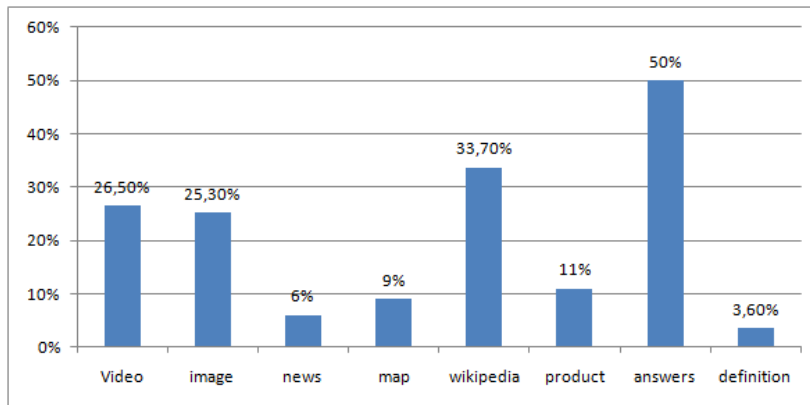


Figure 10.4: Distribution of complementary sources when the primary source is Web search

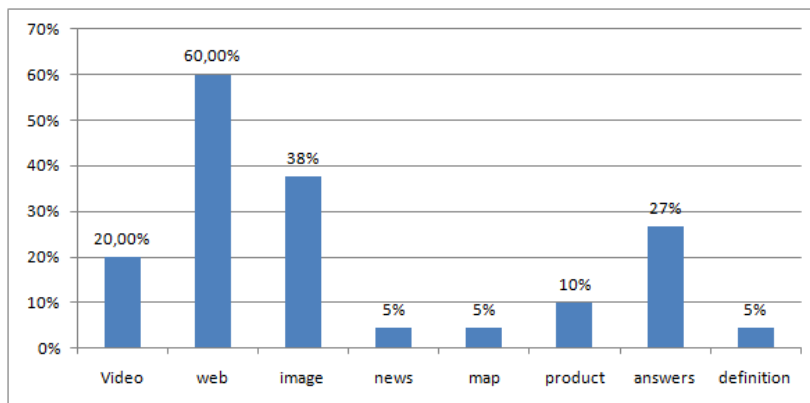


Figure 10.5: Distribution of complementary sources when the primary source is Wikipedia

### 10.3.4 Impact of relevance and query types on evaluation

Our goal now is to compare different evaluation setups for cross-vertical aggregated search. The main question here is whether these setups capture realistically the notion of relevance. We will analyze the impact of the type

of query (short text query versus fixed need) and type of relevance (by intent versus by content).

### Short text query vs. fixed information need

Table 10.2 shows that fixing the information need decreases the amount and diversity of relevant sources. This can be seen comparing results from task 1 and task 3 and those of task 2 and task 4. For instance, for the query “*Hamilton County*”, the information need was “*I want the location of Hamilton County*”. For this query, in *U1* we can choose images and Wikipedia as relevant. Once fixed the information need these sources become irrelevant.

### Relevance by intent versus relevance by content

Our aim here is to discuss the differences of results when considering relevance by intent or by content. We found that both types of relevance assessment contribute in identifying relevant sources. However, there are sources that are considered relevant by intent and irrelevant by content and vice-versa. Concretely, in 7.2% of cases a source was considered relevant by intent (by at least one participant) in *U1* and irrelevant (by content) in *U2* (from all participants). In 26.3% of cases, a source was found relevant in *U2* and irrelevant in *U1*. In 7.3% of cases, a source was found relevant in *U3* and irrelevant in *U4*. In 19.6% of cases, a source was found relevant in *U4* and irrelevant in *U3*. **Clearly, participants could identify more relevant sources by content than by intent.**

The above has some explications. For many queries, participants did not have enough knowledge or they could not imagine any relevant information in some specific source. Viewing results helps them finding additional useful information.

On the other hand, some sources can be considered irrelevant by content because they provide no relevant results, even if the participant could imagine something relevant by intent (*U1* and *U3*). There can be at least two explications. There exists nothing relevant for that query or the source did not work well.

In order to fully understand these results, we analyzed the participants agreement as well as time they spent assessing the results. This analysis is described below.

### Participants agreement and time

We define session duration to be the average time spent by one participant to assess a query for all sources. The session durations are shown in table 10.4 with respect to each task. As we can expect, evaluation of relevance by intent is 5 to 8 times faster than evaluation by content. It is 5 times faster

when the query is just short text. It is 8 times faster when the information need is fixed. We also observe that fixing the information need makes assessments faster.

We also computed participant agreement for each task of the study. We use Fleiss' Kappa  $k$ , a measure for assessing the reliability between a fixed number of participants. It is used in alternative to Cohen's Kappa which is used to measure agreement between exactly two participants. Depending on the range this measure falls, we can classify agreement into 6 classes: poor agreement ( $k < 0$ ), slight agreement ( $k \in [0, 0.2]$ ), fair agreement ( $k \in [0.21, 0.4]$ ), moderate agreement ( $k \in [0.41, 0.6]$ ), substantial agreement ( $k \in [0.61, 0.8]$ ) and almost perfect agreement ( $k \in [0.81, 1]$ ). Results are shown in table 10.4. The first task of the study falls in the range of fair agreement. The others fall in the range of moderate agreement. We can see that fixing the information need increases agreement and that agreement is highest for evaluation by content.

The agreement level of our tasks does not affect the validity of results in the previous sections, because low inter-assessor agreement is common in IR evaluation. This has also been recognized in the context of major evaluation campaigns such as TREC, INEX [100].

<b>Study</b>	<b>U1</b>	<b>U2</b>	<b>U3</b>	<b>U4</b>
<b>Session duration (seconds)</b>	25	180	24	105
<b>Inter participant agreement</b>	0.36	0.48	0.46	0.56

Table 10.4: Duration and agreement by task

To conclude, we showed that evaluating by intent can introduce bias in the evaluation process. Participants miss many relevant sources, which they can identify when they access results by content. Evaluating by intent is faster (see table 10.4), but it produces a lower participant agreement.

### 10.3.5 Questionnaires

In this section we analyze the answers to the questionnaires. Some of the observations are quite interesting.

This is a summary of the pre-study questionnaires. All users are familiar to search engines. 43% declare they usually use just one source for their search tasks. 26% use two sources on average and 30% use three or more sources for their search tasks. They are all familiar with vertical search engines and they know 3 or more vertical search engines each.

In the post-questionnaire, we analyze the task difficulty. The second task was considered the most difficult, followed by the forth, the third and the first. We can deduct that the absence of a fixed information need makes the

task more difficult. As well, viewing search results demands a lot of time, though it influences the task difficulty.

We asked specific question concerning the query interpretations, ambiguity and complementary results. 86% of the participants of task 2 found at least one query they could not interpret initially, but in 57% of the cases they could find an interpretation after viewing some search results. 86% found at least one query with more than one interpretation. Half of them could disambiguate these queries. All users answered positively to the question “After this study, do you believe that different search sources can be complementary to each other”.

Finally, we asked the question “Do you believe that an aggregated search system composed of the 9 sources provides more relevant results than Web search engines?”. 20% of the users answered “yes”s, 66% answered “maybe” and 14% answered “no”. We should state that this question is not very coherent as today major search engines implement cross-vertical aggregated search and this may cause some confusion.

## 10.4 Discussion

Our study proves that cross-vertical aggregated search is not just about coloring search results with images and videos. We could identify clear advantages of cross-vertical aggregated search and useful tips for evaluation. Some of the findings were already known in literature, but some others are new.

It was already known that vertical intent is frequently present within queries and this was confirmed in this study. It was already known that cross-vertical aggregated search increases the diversity of relevant search results and this was confirmed in this study. It is though interesting to observe from this study that the vertical search engines do not fail to satisfy the vertical intent i.e. mostly they were selected as relevant by both intent and content.

It is also interesting to investigate the advantages of cross-vertical aggregated search through comparison of the utility of traditional Web search with utility of vertical search engines. We observed that vertical search engines and Web search engines were often relevant at the same time, but mostly they presented different information. Though they were useful at the same time. Furthermore, vertical search engines were frequently found relevant when Web search failed to return relevant results. We can conclude that vertical search engines can play an important role in the new Web search. They answer many queries and they complete Web search.

A further investigation concerns diversity of relevant results. We found that many queries match more than one relevant source. We could identify three reasons to explain the latter. First, some of our queries are ambiguous.



Second, some sources complete each other (return different aspects of the same need). Third, some sources return repeating information. We show that often one source alone is not enough, while multiple sources can complete each other. At our knowledge, there was no focused investigation on the reasons behind the diversity of relevant sources.

We investigated the differences among the 4 search situations trying to derive useful conclusions for the evaluation of *cvAS*. First, we notice that fixing the information need decreases the amount and diversity of relevant sources, but it eases the evaluation task. Less assessment time is needed and inter-assessor agreement is higher. Second, we notice that assessing by intent has some significant drawbacks. Assessors miss interpretations of the query. They can identify more relevant sources when they access search results (relevance by content). The major strength of this kind of evaluation is assessment time which is significantly lower.

Task 1 of our study uses the same configuration as evaluation in [14, 104, 108] (short queries, relevance by intent). Although this form of evaluation is fast, it is also the less realistic. Assessors do not know the real information need and they often miss identifying relevant sources. On the other hand, evaluation as done in [166, 168] is closer to traditional IR evaluation (fixed need, relevance by content). The major drawbacks of this approach is that it is time-consuming and that it depends on the quality of sources.

We believe that the learnings of our study can be useful for future evaluation of *cvAS* approaches.

## 10.5 Conclusions

In this chapter we describe our research on the interest and the evaluation techniques for cross-vertical aggregated search. We considered both short-text queries and fixed need queries. We also investigated for the first time both relevance by intent and relevance by content. In each task of our study, relevance is assessed for 100 queries across 9 sources (vertical searches and Web search).

We could identify advantages of cross-vertical aggregated search which were not assessed before. Furthermore comparing the 4 search situations we could derive useful thoughts on the evaluation of cross-vertical aggregated search.



## Part V

# Part V: Conclusions and future work



## Chapter 11

# Conclusions and future work

### 11.1 Conclusions

The work presented in this thesis concerns aggregated search, which we consider as a third generation of information retrieval approaches. Precisely, we distinguished three broad paradigms in the IR history namely boolean retrieval, ranked retrieval and aggregated retrieval. Our overview of aggregated search approaches shows the many different research directions. Within them we distinguished three important general components namely query dispatching, nugget retrieval and result aggregation. These components are found in different instantiations of aggregated search including approaches from question answering, information extraction, natural language generation, federated search, etc.

Our research falls within two research directions namely: *relational aggregated search* and *cross-vertical aggregated search*. Our contribution can though be split in two parts, one per research direction. We will summarize and conclude on our contribution for each of them.

#### 11.1.1 Relational aggregated search

Relational aggregated search was presented as a new paradigm where retrieval targets information nuggets and their relations, while result is assembled taking into account for relations. In this context, we investigated on 3 information nuggets (classes, instances and attributes) and their explicit relations. Our contribution includes a large-scale approach for attribute retrieval, a weight-based approach for result aggregation, a large-scale extraction method for lists of named entities and four relational aggregated search applications (prototypes):

**Attribute retrieval:** We presented an approach for Web-scale attribute retrieval using HTML tables. Our attribute retrieval approach relies on 3 recall-oriented filters and relevance ranking. Results are promising both in terms of precision and recall.

Our approach was tested for 3 different attribute retrieval problems: instance attribute retrieval, class attribute retrieval, reinforced attribute retrieval. In the first case, we retrieve attributes for a given instance. In the second case, we retrieve attributes at class level. Our attribute retrieval approach is shown to have a high recall and good precision for both problems. Experiments on reinforced attribute retrieval show that using similar instances of the same class to reinforce instance attribute retrieval improves significantly retrieval performance.

We also validated the utility of our filters: relational filter, header filter and attribute line filter. In our experimental setup all filters prove their effectiveness. They are shown to have a high recall over positives and to filter out a reasonable amount of useless data. As well, the application of filters is shown to have a positive impact on attribute retrieval. This corresponds to better precision without an important loss in recall.

To resume, our attribute retrieval approach has been shown different advantages. To start with, it has a high coverage i.e. it can be applied to many instances. Second, it works at Web scale and it has a high recall in terms of retrieved relevant attributes. Its recall is significantly higher than quality sources such as DBPedia and Wikipedia. Furthermore, it outperforms state of the art techniques for the same purpose.

**Result aggregation:** In addition, we investigated on result aggregation for relational aggregated search. Here, we proposed a weight-based framework for result aggregation with a focus on the construction of tabular results for class queries. This framework contains weights for both instances and attributes. The experimental framework confirms that weight-based ranking of instances and attributes has an important effect on the final quality of results. As well, we showed that result aggregation in relational aggregated search has some unique issues that are not met in traditional ranked retrieval.

**Lists of instances:** We also proposed an investigation on large-scale extraction of lists of instances which is supposed to assist relational aggregated search. In particular, we propose an alternative approach to extract lists of instances without relying on class acquisition.

The extraction source was HTML lists from the web. Using a random sample of HTML lists and we showed that about 8.25% of the HTML lists in the dataset were lists of class instances. If our estimation is validated and can be generalized to the Web, using an estimation of the size of the Web, we can estimate to have more 892 million lists of named entities of the same type, which corresponds to billions of named entities.

The identification of such a huge collection of named entities of the same type is only one part of the contribution. Further analysis on our assessed lists is used to understand which features are more discriminative for their

extraction. In addition, we proposed an automatic extraction technique for lists of instances that relies on these features.

**Prototypes:** We also showed 4 prototypes for relational aggregated search that we build using this work. The first three are dedicated to just one query type respectively to attribute queries, instance queries and class queries. The forth prototype allows all three types of queries and answers each type of query with the appropriate designated solution. Within these approaches, we have also integrated passage and image retrieval which fit naturally in the returned results. These approaches show that relational aggregated search can be instantiated in different forms. At the same time, it proved that our techniques provide encouraging results.

To summarize, we introduced relational aggregated search as a promising research direction within aggregated search. We presented high-recall approaches to extract/retrieve relational data. Their relations were shown useful to build aggregated search results. Our experiments and prototypes prove that relational aggregated search can be instantiated at large-scale with promising results.

### 11.1.2 Cross-vertical aggregated search

In the context of cross-vertical aggregated search, we identified clear advantages of cross-vertical aggregated search and useful tips for evaluation. Some of the findings were already known in literature, but some others are new. For instance, it was already known that vertical intent is frequently present within queries and this was confirmed in this study. It was already known that cross-vertical aggregated search increases the diversity of relevant search results and this was confirmed in this study.

Through our assessments, we compared the utility of traditional Web search with utility of vertical search engines. We observed that vertical search engines and Web search engines were often relevant at the same time, but mostly they presented different information. In these cases, we can say that they are both useful and non redundant. Furthermore, vertical search engines were frequently found relevant when Web search failed to return relevant results. We can conclude that vertical search engines can play an important role in the context of Web search. They answer many queries and they often complete traditional Web search.

A further investigation concerns diversity of relevant results. We found that many queries match more than one relevant source. We identified three reasons to explain this. First, some of our queries are ambiguous. Second, some sources complete each other (return different aspects of the same need). Third, some sources return repeating information. We show that often one

source alone is not enough, while multiple sources can complete each other. At our knowledge, there was no focused investigation on the reasons behind the diversity of relevant sources.

We investigated the differences among the 4 relevance assessment setups trying to derive useful conclusions for the evaluation of *cvAS*. First, we notice that fixing the information need decreases the amount and diversity of relevant sources, but it eases the evaluation task. Less assessment time is needed and inter-assessor agreement is higher. Second, we notice that assessing by intent has some important drawbacks. Assessors miss interpretations of the query and they have low inter-assessor agreement. The major strength of this kind of evaluation is assessment time which is significantly lower.

## 11.2 Future work

Both relational aggregated search and cross-vertical aggregated search are new research areas, though there is a lot of work to be done in research. Here, we list from our perspective some important future research in short and long term.

### 11.2.1 Relational aggregated search

From our perspective the most important research directions in relational aggregated search concern relation retrieval and result aggregation. Indeed, the more we can retrieve relations, the more we can answer and the better we can assemble search results.

As shown in this work, relation retrieval is crucial in this type of search. We have shown how precision oriented attribute extraction can be turned into recall oriented attribute retrieval which enables focused and assembles answers. By analogy, we can instantiate relation retrieval for instances, passages, images, etc. For instance, given a class query we can retrieve its instances which includes an “is instance of” relation. Similarly for instance queries we can retrieve passages, images, etc. which can correspond to a “is about” relation.

The more we can relate, the more we can aggregate. Indeed, we have seen through examples that passages, images and attributes can be assembled in various formats in the relational aggregated search framework. We believe that in this direction there is a lot of work to do.

Another important issue that should be object of research in future is the evaluation of these approaches. In fact, information retrieval is scrupulous when it comes to evaluation and relational aggregated search should not represent an exception.

In the short term, we will tackle some of the issues closely related with what we have presented in this work. We will list some of them:



We believe that attribute retrieval from HTML tables can still be improved.

We will soon start with more focused research on attribute values.

We need here to deal with attribute values of different types, missing values, multi-value attributes. We believe that this work will affect the overall performance on both attribute names and values.

We also believe that attribute retrieval can take benefit from attribute acquisition techniques that extract from pure text. We believe that their recall is lower than the recall of HTML tables, but we might find there attributes which are not naturally met in HTML tables.

We also foresee to extend our work with passages, images and other types of information nuggets. In particular, we can draw some analogies between attributes, passages and aspects. Attributes and passages can both have a name (title for passages) and value (text body for passages). We believe that the combination of attributes and passages provides an aspect oriented representation of information, which we want to exploit for empowering relational aggregated search.

Instance retrieval is one of the problems we list as important for relational aggregated search, although we do not deal with in this work. This is mostly due to insufficient time. For future work, we would like to adapt the approach of Heart [74] for online instance retrieval. We also would like to use contextual text around similar named entities to capture the class name.

In this work, we focused on tabular results, although we presented other ways to assemble relational aggregated search results through our prototypes. For future work, we would like to focus on instance queries to check if we can go beyond listing approaches. As well, we would like to investigate on query interpretation, because we need to capture the query type before triggering retrieval and result aggregation.

### 11.2.2 Cross-vertical aggregated search

From our point of view, research on cross-vertical aggregated search is its beginnings. The potential of vertical searches remains to be explored and there are many vertical search engines that are not exploited. Future approaches should take advantage from the specific advantages of each source as well as from their combination. From this perspective, we list two important research directions.

First, it is important to enable flexible and sound techniques that allow integrating results from more sources taking into account for their specificities not only in terms of query matching but also in terms of utility. The utility of each source can be different in terms of focus, diversity, service,

etc. Integrating a focused/specific result might not have the same benefit with a broad result; integrating irrelevant images might be less harmful than integrating irrelevant textual information, etc. Probably a taxonomy for vertical search engines would help in this direction.

Second, two or more sources can be helpful at the same time as it was shown in this work. The utility of multiple sources at once remains to be explored. Are there some type of queries which demand multiple sources? Are there sources which are more likely to be complementary to a given source? Can we go beyond ranking or block ranking? Ideally, the combination of results from different sources should provide a complete and well-organized answer.

In the short term, we will focus on the following issues:

We are currently investigating on approaches for flexible ranking of search results where results can be grouped in blocks of variable size. Current techniques demand re-training the ranking function when the block size is changed. In the near future, we will finish our experiments on flexible result ranking. This will involve different learning to rank techniques, variable block sizes. We aim to prove that through few training data, we can provide flexible result ranking which can be varied through time. Our experiments will also include a simple setup for relevance assessments.

We are also investigating towards contextual uses of cross-vertical aggregated search. This is different from existing work which is designed for Web search. Our intuition is that different vertical search engines can be combined to satisfy context specific tasks where there is not necessarily a primary source.

Last but not least within our future research directions will be evaluation. After our study with 4 relevance assessment setups, we would like to continue our investigation on evaluation issues. Our focus will be on different levels of relevance and their assessment for different tasks such as source selection and result ranking.

# Appendix A

## Datasets

This chapter contains the queries used to test attribute retrieval and the queries used in the evaluation-oriented study for cross-vertical aggregated search.

### A.1 Dataset for attribute retrieval evaluation

The attribute retrieval approach was tested using 200 queries corresponding to 20 classes with 10 instances each. In this section we will provide these queries. Below, there is the list of classes:

*rock bands, laptops, American universities, hotels, software, British Army generals, chancellors of Germany, American films, IR articles, SLR cameras, novels, nirvana songs, nissan vehicles, programmable calculators, countries, drugs, companies, cities, painters, mobile phones*

We will now enumerate the instances by class. The class name (in bold) follows the list of instances (in italic).

**Rock bands:** *Alice in chains, Red Hot Chili Peppers, Aerosmith, Blink 182, Green Day, Placebo, The verve, Good Charlotte, Depeche Mode, U2*

**Laptops:** *Apple MacBook Air, Acer TravelMate 8172, Toshiba satellite C650, Lenovo ThinkPad T410, HP Compaq Presario CQ56Z, Dell Inspiron 15, Sony Vaio VPCZ138GG, Toshiba satellite L450, Apple MacBook, Acer Aspire 1425P*

**American universities:** *Harvard University, Yale University, Stanford University, Boston University, Massachusetts Institute of Technology, Carnegie Mellon University, Cornell University, Duke University, Emory University, Ohio State University*

**Hotels:** *Commodore Hotel New York Glasgow International Hilton Hotel The George Hotel Edinburgh Detroit Riverside Hotel Denver Marriott City Center Chicago Beach Hotel Hyatt Regency Chicago Plaza Hotel The Scotsman Hotel Ritz-Carlton Denver*

**Software:** *ObjectVision, openFIRST, OpenPilot, PrintMaster, PsyToolkit, QuickOffice, ArgoUML, SoftwareGR, Ultamatix, Apache Beehive*

**British Army generals:** *Henry Shrapnel, Sir William Williams, John Moore, Henry Clinton, Thomas Gage, William Ponsonby, William Boog Leishman, James Inglis Hamilton, Gerald Grosvenor, Stewart Menzies*

**Chancellors of Germany:** *Otto von Bismarck, Helmut Kohl, Angela Merkel, Georg Michaelis, Philipp Scheidemann, Lutz Graf Schwerin von Krosigk, Hans Luther, Gerhard Schroder, Kurt Georg Kiesinger, Prince Maximilian of Baden*

**American films:** *Escape from New York, Mulholland Drive, American Psycho 2, Death Wish II, Paths of Glory, Taxi Driver, Crossfire, From Here to Eternity, The Big Fisherman, In Old Arizona*

**IR papers:** *Learning to Rank for Information Retrieval, Opinion Mining and Sentiment Analysis, Overview of the TREC 2004 question answering track, Incremental Clustering of Newsgroup Articles, Extracting Product Features and Opinions from Reviews, Faceted Metadata for Image Search and Browsing, Detecting geographic locations from web resources, Automatic Information Organization and Retrieval, Performance prediction of data fusion for information retrieval, Data fusion with estimated weights*

**SLR cameras:** *Minolta Maxxum 7000, Canon New F-1, Olympus OM-2, Canon EOS 100, Pentax MZ-S, Pentax MV 1, Agfa Ambiflex, Pentax Super-A, Canon EOS, Canon AE-1*

**Novels:** *House of Leaves, The Blind Assassin, Harry Potter and the Goblet of Fire, The Amber Spyglass, The Summons, Revelation Space, Lord Brocktree, Tainted Blood, The Deadly Hunter, He Shall Thunder in the Sky*

**Nirvana songs:** *Aero Zeppelin, Anorexorcist, In the Pines, Heart-Shaped Box, Sliver, Smells Like Teen Spirit, Come as You Are, All Apologies, Aneurysm, Drain You*

**Nissan vehicles:** *Nissan Rogue, Nissan Langley, Nissan 240SX, Nissan Serena, Nissan Maxima, Nissan Almera, Nissan Primera, Nissan Qashqai, Nissan Quest, Nissan Gloria*

**Programmable calculators:** *HP-55, Remington Rand 409, Harvard Mark I, Sharp PC-1403, Casio FA-1, IBM 608, Monroe Epic, TI-68, UNIVAC 120, Sharp EL-5120*

**Countries:** *Albania, Algeria, Tunisia, Turkey, China, Canada, South Africa, Saudi Arabia, Egypt, Australia*

**Drugs:** *Diazaborine, Acetasol, Isopropamide, Nexium, Prevacid, Prilosec, Pantoprazole, Rabeprazole, Tamiflu, Paracetamol*

**Companies:** *Dubai Holding, Acromas Holdings, Europapress Holding, British Airways, Air France, City Bank, Wilkinson Sword, Royal Bank of Scotland Group, General Motors, The Emirates Group*

**Cities:** *Abu Dhabi, Bruxelles, London, Tirana, Tunis, Istanbul, Rome, Shanghai, Sidney, Los Angeles*

**Painters:** *Sandro Botticelli, Raffaello Sanzio, Michelangelo Buonarroti, Giotto di Bondone, Pablo Picasso, Salvador Dali, Wassily Kandinsky, Claude Monet, Vincent van Gogh, Leonardo da Vinci*

**Mobile phones:** *Nokia e72, LG A130, Blackberry Torch, iPhone 2, Sony Ericsson Xperia, Nokia e7, HTC Touch, Blackberry Storm, LG Xenon, Samsung Galaxy S*

## A.2 Dataset for the cross-vertical aggregated search study

In this section, we list the queries that were used in the study about interest and evaluation of cross-vertical aggregated search. We recall that these queries were samples from the Million Query Track. The list of queries is as follows:

*super novae, eczema pictures, milestone herbicide, free nero 8 downloads, work place harassment, 49 cfr 391, single mothers, finished basements, inauguration of james madison, my web tech, social security disability and hiv, books in braille, iowa va hospital, wet swimsuit, saami standards, capital one canada, pay traffic tickets in nyc, bannack mt, nutri shop, valley of the drums louisville, mordecai house, bearshare, shih tzu for sale, men taking showers, hamilton county website, oasas ny state, dry rub, promethazine, idaho gas tax, used car parts, newspapers, quality of life measures, drug interactions, ford, bellingham, rio song, john z delorean, malton art gallery, orange county health department, poor circulation in the foot, ryan matthews, cartoon sharks, homecomings financial, byrd, gad65 auto antibody, stage 1 lung cancer, 180*

*poem, kirby vacuum, jack wang, mitigation, seguro social puerto rico, satop classes mo, metre, cobb water system, musicaparaguaya, friendship questionnaires, night vision video camera, playskool bus, west valley city zoning, stomach pain causes, whole sale tire, wormian bones, tang soo do, stages of photosynthesis, dwarven forge, tulalip jobs, time and date, tim burton alice in wonderland, orange county new york yellow pages, metal spray, directions, sts columbia, free answers to tax questions, wichita state university, shattuck national bank, seth unger, crape myrtle, beaches in ibiza, celebrities flashing their knickers, communicable diseases, educational publishing, the family act, so you want to be a supermodel, the bahamas drinking age, free ringtones, trustees, the mechanical universe, calculating fers retirement, transfemoral, state of michigan building codes, microsoft powerpoint download free, twilight dress up games, off we go into the wild blue, positional words games, broken bones, middle school algebra, remichel, fat granny, buchanan auto auctions, ireland vacation*

# Bibliography

- [1] <http://www.nist.gov/tac/tracks/2008/summarization/>, 2008. TAC Summarization Track 2008.
- [2] <http://academic.research.microsoft.com>, 2010. accessed in april 2010.
- [3] <http://shopping.yahoo.com/>, 2010. accessed in april 2010.
- [4] <http://www.123people.com>, 2010. accessed in april 2010.
- [5] <http://www.bing.com/shopping>, 2010. accessed in april 2010.
- [6] Pal Aditya and Kawale Jaya. Leveraging query association in federated search. In *SIGIR 2008 Workshop on aggregated search*.
- [7] Eugene Agichtein and Luis Gravano. Snowball: extracting relations from large plain-text collections. In *DL '00: Proceedings of the fifth ACM conference on Digital libraries*, pages 85–94, New York, NY, USA, 2000. ACM.
- [8] Rakesh Agrawal, Sreenivas Gollapudi, Alan Halverson, and Samuel Ieong. Diversifying search results. In *WSDM '09: Proceedings of the Second ACM International Conference on Web Search and Data Mining*, pages 5–14, New York, NY, USA, 2009. ACM.
- [9] Enrique Alfonseca, Marius Pasca, and Enrique Robledo-Arnuncio. Acquisition of instance attributes via labeled and related instances. In *SIGIR '10: Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 58–65, New York, NY, USA, 2010. ACM.
- [10] Abdulrahman Almuhareb and Massimo Poesio. Attribute-based and value-based clustering: An evaluation. In *In EMNLP '04, ACL*, pages 158–165, 2004.
- [11] Marilisa Amoia and Claire Gardent. Adjective based inference. In *Proceedings of the Workshop KRAQ'06 on Knowledge and Reasoning for Language Processing*, KRAQ '06, pages 20–27, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

- [12] Jaime Arguello, Fernando Diaz, and Jamie Callan. Learning to aggregate vertical results into web search results. In *Conference on Information and Knowledge Management, CIKM*, pages 201–210, 2011.
- [13] Jaime Arguello, Fernando Diaz, Jamie Callan, and Ben Carterette. A methodology for evaluating aggregated search results. In *Proceedings of the 33rd European conference on Advances in information retrieval, ECIR’11*, pages 141–152, 2011.
- [14] Jaime Arguello, Fernando Diaz, Jamie Callan, and Jean-Francois Crespo. Sources of evidence for vertical selection. In *SIGIR ’09: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 315–322, New York, NY, USA, 2009. ACM.
- [15] Jaime Arguello, Fernando Diaz, and Jean-François Paiement. Vertical selection in the presence of unlabeled verticals. In *SIGIR ’10: Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 691–698, New York, NY, USA, 2010. ACM.
- [16] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. Dbpedia: A nucleus for a web of open data. In *ISWC/ASWC*, pages 722–735, 2007.
- [17] Yonatan Aumann, Ronen Feldman, Yair Liberzon, Benjamin Rosenfeld, and Jonathan Schler. Visual information extraction. *Knowl. Inf. Syst.*, 10:1–15, July 2006.
- [18] Thi Truong Avrahami, Lawrence Yau, Luo Si, and Jamie Callan. The fedlemur project: Federated search in the real world. *JASIST*, 57(3):347–358, 2006.
- [19] K. Balog, A. P. de Vries, P. Serdyukov, P. Thomas, and T. Westerveld. Overview of the trec 2009 entity track. In *TREC 2009 Working Notes*. NIST, November 2009.
- [20] K. Balog, A. P. de Vries, P. Serdyukov, P. Thomas, and T. Westerveld. Overview of the TREC 2009 entity track. In *Proceedings of the Eighteenth Text REtrieval Conference (TREC 2009)*. NIST, February 2010.
- [21] K. Balog, P. Serdyukov, and A. P. de Vries. Overview of the TREC 2010 entity track. In *Proceedings of the Nineteenth Text REtrieval Conference (TREC 2010)*. NIST, February 2011.



- [22] SearchChannel/Slack Barshinger. The emerging opportunity in vertical search: A review of niche-oriented search engines and directories. Technical report, May 2006.
- [23] Brian T. Bartell, Garrison W. Cottrell, and Richard K. Belew. Automatic combination of multiple ranked retrieval systems. In *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 173–181, New York, NY, USA, 1994. Springer-Verlag New York, Inc.
- [24] Senjuti Basu Roy, Sihem Amer-Yahia, Ashish Chawla, Gautam Das, and Cong Yu. Constructing and exploring composite items. In *Proceedings of the 2010 international conference on Management of data, SIGMOD '10*, pages 843–854, New York, NY, USA, 2010. ACM.
- [25] Mikhail Bautin and Steven Skiena. Concordance-based entity-oriented search. *Web Intelli. and Agent Sys.*, 7:303–319, December 2009.
- [26] Mustapha Baziz, Mohand Boughanem, Yannick Loiseau, and Henri Prade. Fuzzy Logic and Ontology-based Information Retrieval. In Paul Wang, Da Ruan, Etienne Kerre, and , editors, *Studies in Fuzziness and Soft Computing*, volume 215/2007, pages 193–218. Springer, <http://www.springerlink.com>, 2007.
- [27] Nicholas J. Belkin and W. Bruce Croft. Information filtering and information retrieval: Two sides of the same coin? *Commun. ACM*, 35(12):29–38, 1992.
- [28] Kedar Bellare, Partha Pratim Talukdar, Giridhar Kumaran, O Pereira, Mark Liberman, Andrew McCallum, and Mark Dredze. Lightly-supervised attribute extraction for web search. In *Proceedings of Machine Learning for Web Search Workshop, NIPS 2007*, 2007.
- [29] Ori Ben-Yitzhak, Nadav Golbandi, Nadav Har’El, Ronny Lempel, Andreas Neumann, Shila Ofek-Koifman, Dafna Sheinwald, Eugene Shekita, Benjamin Sznajder, and Sivan Yogev. Beyond basic faceted search. In *WSDM '08: Proceedings of the international conference on Web search and web data mining*, pages 33–44, New York, NY, USA, 2008. ACM.
- [30] Mohand Boughanem, Claude Chrisment, and Chantal Soulé-Dupuy. Query Modification based on relevance back-propagation in adhoc environnement. *Information Processing & Management*, 35:121–139, avril 1999.
- [31] Mohand Boughanem and Jacques Savoy. *Recherche d’information : Etat des lieux et perspectives*. Hermes Science Publications, April 2008.

- [32] Ourdia Boudighaghen, Lynda Tamine, and Mohand Boughanem. Dynamically Personalizing Search Results for Mobile Users. In *Flexible Query Answering (FQAS)*, volume 5822/2009, pages 99–110, 2009.
- [33] Bert R. Boyce. Beyond topicality : A two stage view of relevance and the retrieval process. *Inf. Process. Manage.*, 18(3):105–109, 1982.
- [34] Sergey Brin. Extracting patterns and relations from the world wide web. In *WebDB '98: Selected papers from the International Workshop on The World Wide Web and Databases*, pages 172–183, London, UK, 1998. Springer-Verlag.
- [35] Michael J. Cafarella, Michele Banko, and Oren Etzioni. Relational web search. Technical report, University of Washington, 2006.
- [36] Michael J. Cafarella, Alon Y. Halevy, and Nodira Khoussainova. Data integration for the relational web. *PVLDB*, 2(1):1090–1101, 2009.
- [37] Michael J. Cafarella, Alon Y. Halevy, Yang Zhang, Daisy Zhe Wang, and Eugene Wu. Uncovering the relational web. In *WebDB*, 2008.
- [38] Jamie Callan. Distributed information retrieval. In W. Bruce Croft, editor, *Advances in Information Retrieval*, pages 235–266. Kluwer Academic Publishers, Dordrecht, 2000.
- [39] Jaime Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 335–336, New York, NY, USA, 1998. ACM.
- [40] Ben Carterette, Virgil Pavlu, Evangelos Kanoulas, Javed A. Aslam, and James Allan. Evaluation over thousands of queries. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 651–658, New York, NY, USA, 2008. ACM.
- [41] Chia-Hui Chang, Mohammed Kayed, Moheb Ramzy Girgis, and Khaled F. Shaalan. A survey of web information extraction systems. *IEEE Trans. on Knowl. and Data Eng.*, 18:1411–1428, October 2006.
- [42] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: a library for support vector machines. Technical report, 2001.
- [43] Hsin-Hsi Chen, Shih-Chung Tsai, and Jin-He Tsai. Mining tables from large scale html texts. In *Proceedings of the 18th conference on Computational linguistics - Volume 1, COLING '00*, pages 166–172, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

- [44] Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information, and lexicography. In *Proceedings of the 27th annual meeting on Association for Computational Linguistics*, ACL '89, pages 76–83, Stroudsburg, PA, USA, 1989. Association for Computational Linguistics.
- [45] Charles L. A. Clarke, Maheedhar Kolla, Gordon V. Cormack, Olga Vechtomova, Azin Ashkan, Stefan Büttcher, and Ian MacKinnon. Novelty and diversity in information retrieval evaluation. In *SIGIR*, pages 659–666, 2008.
- [46] Charles L.A. Clarke, Nick Craswell, and Ian Soboroff. Overview of the trec 2009 web track. Technical report, 2010.
- [47] Paul Clough, Mark Sanderson, Murad Abouammoh, Sergio Navarro, and Monica Lestari Paramita. Multiple approaches to analysing query diversity. In *SIGIR*, pages 734–735, 2009.
- [48] William W. Cohen and Wei Fan. Web-collaborative filtering: recommending music by crawling the web. *Comput. Netw.*, 33(1-6):685–698, 2000.
- [49] Valter Crescenzi, Giansalvatore Mecca, and Paolo Merialdo. Roadrunner: Towards automatic data extraction from large web sites. In *Proceedings of the 27th International Conference on Very Large Data Bases*, VLDB '01, pages 109–118, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.
- [50] Bruce Croft, Donald Metzler, and Trevor Strohman. *Search Engines: Information Retrieval in Practice*. Addison-Wesley Publishing Company, USA, 2009.
- [51] Hoa Tang Dang. Overview of DUC 2006, 2006.
- [52] Kushal Dave, Steve Lawrence, and David M. Pennock. Mining the peanut gallery: opinion extraction and semantic classification of product reviews. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 519–528, New York, NY, USA, 2003. ACM.
- [53] Fernando Diaz. Integration of news content into web results. In *WSDM*, pages 182–191, 2009.
- [54] Fernando Diaz and Jaime Arguello. Adaptation of offline vertical selection predictions in the presence of user feedback. In *SIGIR '09: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 323–330, New York, NY, USA, 2009. ACM.

- [55] Anlei Dong, Yi Chang, Zhaohui Zheng, Gilad Mishne, Jing Bai, Ruiqiang Zhang, Karolina Buchner, Ciya Liao, and Fernando Diaz. Towards recency ranking in web search. In *WSDM '10: Proceedings of the third ACM international conference on Web search and data mining*, pages 11–20, New York, NY, USA, 2010. ACM.
- [56] Miles Efron. Generative model-based metasearch for data fusion in information retrieval. In *Proceedings of the 9th ACM/IEEE-CS joint conference on Digital libraries, JCDL '09*, pages 153–162, New York, NY, USA, 2009. ACM.
- [57] Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S. Weld. Open information extraction from the web. *Commun. ACM*, 51:68–74, December 2008.
- [58] Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Unsupervised named-entity extraction from the web: an experimental study. *Artif. Intell.*, 165(1):91–134, 2005.
- [59] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370, Morristown, NJ, USA, 2005. Association for Computational Linguistics.
- [60] Michael Fleischman, Eduard Hovy, and Abdessamad Echihabi. Offline strategies for online question answering: answering questions before they are asked. In *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 1–7, 2003.
- [61] Shlomo Geva, Jaap Kamps, and Andrew Trotman, editors. *Advances in Focused Retrieval: 7th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2008, Dagstuhl Castle, Germany, December 15-18, 2008. Revised and Selected Papers*. Springer-Verlag, Berlin, Heidelberg, 2009.
- [62] Jeremy Goecks. Nuggetmine: Intelligent groupware for opportunistically sharing information nuggets. In *Proceedings of the International Conference on Intelligent User Interfaces (IUI '02)*, ACM, pages 87–94. Press, 2002.
- [63] William Goffman. A searching procedure for information retrieval. *Information Storage and Retrieval*, 2(2):73–78, 1964.
- [64] Jade Goldstein, Vibhu Mittal, Jaime Carbonell, and Mark Kantrowitz. Multi-document summarization by sentence extraction. In *NAACL-ANLP 2000 Workshop on Automatic summarization*, pages 40–48,

- Morristown, NJ, USA, 2000. Association for Computational Linguistics.
- [65] Luis Gravano, Chen-Chuan K. Chang, Héctor García-Molina, and Andreas Paepcke. Starts: Stanford proposal for internet meta-searching. *SIGMOD Rec.*, 26:207–218, June 1997.
- [66] Luis Gravano, Héctor García-Molina, and Anthony Tomasic. The effectiveness of gloss for the text database discovery problem. In *Proceedings of the 1994 ACM SIGMOD international conference on Management of data*, SIGMOD '94, pages 126–137, New York, NY, USA, 1994. ACM.
- [67] Ohad Greenshpan, Tova Milo, and Neoklis Polyzotis. Autocompletion for mashups. *Proc. VLDB Endow.*, 2(1):538–549, 2009.
- [68] H. P. Grice. Logic and conversation. In P. Cole and J. L. Morgan, editors, *Syntax and Semantics: Vol. 3: Speech Acts*, pages 41–58. Academic Press, San Diego, CA, 1975.
- [69] Ralph Grishman and Beth Sundheim. Message understanding conference-6: a brief history. In *Proceedings of the 16th conference on Computational linguistics*, pages 466–471, Morristown, NJ, USA, 1996. Association for Computational Linguistics.
- [70] A. Gulli and A. Signorini. Building an open source meta-search engine. In *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 1004–1005, New York, NY, USA, 2005. ACM.
- [71] Jiafeng Guo, Gu Xu, Xueqi Cheng, and Hang Li. Named entity recognition in query. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 267–274, New York, NY, USA, 2009. ACM.
- [72] Qi Guo, Lizhu Zhou, Hang Guo, and Jun Zhang. Sesq: A novel system for building domain specific web search engines. In *APWeb*, pages 1173–1176, 2006.
- [73] Shun Hattori, Taro Tezuka, and Katsumi Tanaka. Context-aware query refinement for mobile web search. In *SAINT-W '07: Proceedings of the 2007 International Symposium on Applications and the Internet Workshops*, page 15, Washington, DC, USA, 2007. IEEE Computer Society.
- [74] Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational lin-*

- guistics*, pages 539–545, Morristown, NJ, USA, 1992. Association for Computational Linguistics.
- [75] Marti A. Hearst. Automated discovery of wordnet relations. In *C. Fellbaum, WordNet: An Electronic Lexical Database*, pages 131–153. MIT Press, 1998.
  - [76] Marti A. Hearst and Jan O. Pedersen. Reexamining the cluster hypothesis: Scatter/gather on retrieval results. In *SIGIR*, pages 76–84, 1996.
  - [77] Sascha Hennig and Michael Wurst. Incremental clustering of news-group articles. In *IEA/AIE*, pages 332–341, 2006.
  - [78] Mingqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '04*, pages 168–177, New York, NY, USA, 2004. ACM.
  - [79] Panagiotis G. Ipeirotis. *Classifying and searching hidden-web text databases*. PhD thesis, New York, NY, USA, 2004. AAI3147244.
  - [80] Lei Ji, Jun Yan, Ning Liu, Wen Zhang, Weiguo Fan, and Zheng Chen. Exsearch: a novel vertical search engine for online barter business. In *CIKM '09: Proceeding of the 18th ACM conference on Information and knowledge management*, pages 1357–1366, New York, NY, USA, 2009. ACM.
  - [81] Nitin Jindal and Bing Liu. Mining comparative sentences and relations. In *AAAI'06: proceedings of the 21st national conference on Artificial intelligence*, pages 1331–1336. AAAI Press, 2006.
  - [82] Christopher B. Jones and Ross S. Purves. Geographical information retrieval. In *Encyclopedia of Database Systems*, pages 1227–1231. 2009.
  - [83] Jaap Kamps, Shlomo Geva, and Andrew Trotman. Report on the SIGIR 2008 workshop on focused retrieval. *SIGIR Forum*, 42(2):59–65, 2008.
  - [84] Changsung Kang, Srinivas Vadrevu, Ruiqiang Zhang, Roelof van Zwol, Lluís García Pueyo, Nicolas Torzec, Jianzhang He, and Yi Chang. Ranking related entities for web search queries. In *WWW (Companion Volume)*, pages 67–68, 2011.
  - [85] Rianne Kaptein and Maarten Marx. Focused retrieval and result aggregation with political data. *Inf. Retr.*, 13:412–433, October 2010.

- [86] Marcin Kaszkiel and Justin Zobel. Passage retrieval revisited. In *SIGIR '97: Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 178–185, New York, NY, USA, 1997. ACM.
- [87] Makoto P. Kato, Hiroaki Ohshima, Satoshi Oyama, and Katsumi Tanaka. Query by analogical example: relational search using web search engine indices. In *CIKM '09: Proceeding of the 18th ACM conference on Information and knowledge management*, pages 27–36, New York, NY, USA, 2009. ACM.
- [88] Diane Kelly and Jimmy Lin. Overview of the TREC 2007 question answering task. In *In Text REtrieval Conference 2007*, 2007.
- [89] Lyndon S. Kennedy and Mor Naaman. Generating diverse and representative image search results for landmarks. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 297–306, New York, NY, USA, 2008. ACM.
- [90] Arlind Kopliku. Information Aggregation: When search engines organize your night out. In *INFORSID Forum Jeunes Chercheurs, Toulouse, 26/05/09-29/05/09*, pages 451–452, <http://inforsid.irit.fr>, 2009. INFORSID.
- [91] Arlind Kopliku, Mohand Boughanem, and Karen Pinel-Sauvagnat. Querying by examples (poster). In *Conférence francophone en Recherche d'Information et Applications (CORIA), Sousse, Tunisie, 18/03/2010-20/03/2010*, pages 407–408, <http://www.irit.fr/ARIA>, mars 2010. Association Francophone de Recherche d'Information et Applications (ARIA).
- [92] Arlind Kopliku, Mohand Boughanem, and Karen Pinel-Sauvagnat. Mining the Web for lists of Named Entities (short paper). In *Conférence francophone en Recherche d'Information et Applications (CORIA), Avignon, 16/03/2011-18/03/2011*, pages 113–120, <http://www.irit.fr/ARIA>, mars 2011. Association Francophone de Recherche d'Information et Applications (ARIA).
- [93] Arlind Kopliku, Mohand Boughanem, and Karen Pinel-Sauvagnat. Towards a framework for attribute retrieval. In *Conference on Information and Knowledge Management, CIKM*, pages 515–524, 2011.
- [94] Arlind Kopliku, Firas Damak, Karen Pinel-Sauvagnat, and Mohand Boughanem. Interest and Evaluation of Aggregated Search (regular paper). In *IEEE/WIC/ACM International Conference on Web Intelligence, Lyon, 22/08/2011-27/08/2011*, pages 154–161, <http://www.acm.org/>, 2011. ACM.

- [95] Arlind Kopliku, Karen Pinel-Sauvagnat, and Mohand Boughanem. Aggregated search: Potential, issues and evaluation. Technical report, Institut de Recherche en Informatique de Toulouse, 2009.
- [96] Arlind Kopliku, Karen Pinel-Sauvagnat, and Mohand Boughanem. Attribute retrieval from relational web tables. In *Symposium on String Processing and Information Retrieval, SPIRE*, pages 117–128, 2011.
- [97] Arlind Kopliku, Karen Pinel-Sauvagnat, and Mohand Boughanem. Retrieving attributes using web tables. In *Joint Conference on Digital Libraries JDCL*, pages 397–398, 2011.
- [98] Ines Krichen, Arlind Kopliku, Karen Pinel-Sauvagnat, and Mohand Boughanem. Une approche de recherche d’attributs pertinents pour l’agrégation d’information (regular paper). In *INFormatique des Organisations et Systemes d’Information et de Decision (INFORSID), Lille, 24/05/2011-26/05/2011*, pages 385–400, <http://inforsid.irit.fr/>, mai 2011. Association INFORSID.
- [99] Mounia Lalmas. Advanced topics on information retrieval. chapter Aggregated search. Springer, 2011.
- [100] Birger Larsen, Saadia Malik, and Anastasios Tombros. Focused access to xml documents. chapter A Comparison of Interactive and Ad-Hoc Relevance Assessments, pages 348–358. Springer-Verlag, Berlin, Heidelberg, 2008.
- [101] Jongwuk Lee, Seung-won Hwang, Zaiqing Nie, and Ji-Rong Wen. Query result clustering for object-level search. In *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1205–1214, New York, NY, USA, 2009. ACM.
- [102] Joon Ho Lee. Analyses of multiple evidence combination. In *Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '97*, pages 267–276, New York, NY, USA, 1997. ACM.
- [103] Michael S. Lew, Alberto Del Bimbo, and Erwin M. Bakker, editors. *Proceedings of the 1st ACM SIGMM International Conference on Multimedia Information Retrieval, MIR 2008, Vancouver, British Columbia, Canada, October 30-31, 2008*. ACM, 2008.
- [104] Xiao Li, Ye-Yi Wang, and Alex Acero. Learning query intent from regularized click graphs. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 339–346, New York, NY, USA, 2008. ACM.



- [105] Girija Limaye, Sunita Sarawagi, and Soumen Chakrabarti. Annotating and searching web tables using entities, types and relationships. *Proc. VLDB Endow.*, 3:1338–1347, September 2010.
- [106] Ling Lin, Gang Li, and Lizhu Zhou. Meta-search based web resource discovery for object-level vertical search. In *WISE*, pages 16–27, 2006.
- [107] King-Lup Liu, Weiyi Meng, Jing Qiu, Clement Yu, Vijay Raghavan, Zonghuan Wu, Yiyao Lu, Hai He, and Hongkun Zhao. Allinonenews: development and evaluation of a large-scale news metasearch engine. In *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 1017–1028, New York, NY, USA, 2007. ACM.
- [108] Ning Liu, Jun Yan, and Zheng Chen. A probabilistic model based approach for blended search. In *WWW '09: Proceedings of the 18th international conference on World wide web*, pages 1075–1076, New York, NY, USA, 2009. ACM.
- [109] Xiaoyong Liu and W. Bruce Croft. Passage retrieval based on language models. In *Proceedings of the eleventh international conference on Information and knowledge management, CIKM '02*, pages 375–382, New York, NY, USA, 2002. ACM.
- [110] Craig Macdonald. The voting model for people search. *SIGIR Forum*, 43:73–73, June 2009.
- [111] Craig Macdonald and Ian Soboroff. Overview of the trec 2008 blog track. In *TREC*, 2008.
- [112] Christopher Manning, Prabhakar Raghavan, and Heinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.
- [113] M Manoj and Elisabeth Jacob. Information retrieval on internet using meta-search engines: A review. *Journal of Scientific & Industrial Research*, 67:739–746, 2008.
- [114] Kevin S. McCurley. Geospatial mapping and navigation of the web. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 221–229, New York, NY, USA, 2001. ACM.
- [115] Kathleen McKeown, Rebecca J. Passonneau, David K. Elson, Ani Nenkova, and Julia Hirschberg. Do summaries help? In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 210–217, New York, NY, USA, 2005. ACM.

- [116] Kathleen R. McKeown. *Text generation: using discourse strategies and focus constraints to generate natural language text*. Cambridge University Press, New York, NY, USA, 1985.
- [117] Xiaofeng Meng, Haiyan Wang, Dongdong Hu, and Chen Li. A supervised visual wrapper generator for web-data extraction. In *Proceedings of the 27th Annual International Conference on Computer Software and Applications, COMPSAC '03*, pages 657–, Washington, DC, USA, 2003. IEEE Computer Society.
- [118] Xiaofeng Meng, Haiyan Wang, Dongdong Hu, and Chen Li. A supervised visual wrapper generator for web-data extraction. In *COMP-SAC '03: Proceedings of the 27th Annual International Conference on Computer Software and Applications*, page 657, Washington, DC, USA, 2003. IEEE Computer Society.
- [119] Mark Montague and Javed A. Aslam. Condorcet fusion for improved retrieval. In *Proceedings of the eleventh international conference on Information and knowledge management, CIKM '02*, pages 538–548, New York, NY, USA, 2002. ACM.
- [120] Tatsunori Mori and Takuro Sasaki. Information gain ratio meets maximal marginal relevance: A method of summarization for multiple documents. In *Proceedings of the Third NTCIR Workshop*, 2003.
- [121] Véronique Moriceau and Xavier Tannier. Fidji: using syntax for validating answers in multiple documents. *Inf. Retr.*, 13:507–533, October 2010.
- [122] David Mountain and Andrew Macfarlane. Geographic information retrieval in a mobile environment: evaluating the needs of mobile individuals. *Journal of Information Science*, 33(5):515–530, 2007.
- [123] Vanessa Murdock and Mounia Lalmas. Workshop on aggregated search. *SIGIR Forum*, 42(2):80–83, 2008.
- [124] Mor Naaman, Yee Jiun Song, Andreas Paepcke, and Hector Garcia-Molina. Assigning textual names to sets of geographic coordinates. *Computers, Environment and Urban Systems*, 30(4):418–435, 2006.
- [125] Zaiqing Nie, Yunxiao Ma, Shuming Shi, Ji-Rong Wen, and Wei-Ying Ma. Web object retrieval. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 81–90, New York, NY, USA, 2007. ACM.
- [126] Zaiqing Nie, Ji-Rong Wen, and Wei-Ying Ma. Object-level vertical search. In *CIDR*, pages 235–246. [www.crdrrdb.org](http://www.crdrrdb.org), 2007.

- [127] Shiyen Ou and Christopher S. G. Khoo. Aggregating search results for social science by extracting and organizing research concepts and relations. In *SIGIR 2008 Workshop on aggregated search*.
- [128] Satoshi Oyama, Kenichi Shirasuna, and Katsumi Tanaka. Identification of time-varying objects on the web. In *JCDL '08: Proceedings of the 8th ACM/IEEE-CS joint conference on Digital libraries*, pages 285–294, New York, NY, USA, 2008. ACM.
- [129] Patrick Pantel, Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu, and Vishnu Vyas. Web-scale distributional similarity and entity set expansion. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 938–947, Singapore, August 2009. Association for Computational Linguistics.
- [130] Cécile Paris, Stephen Wan, and Paul Thomas. Focused and aggregated search: a perspective from natural language generation. *Information Retrieval Journal*, 44(3), 2010.
- [131] Cecile Paris and Nathalie Collineau. Scifly: tailored corporate brochures on demand. Technical report, CSIRO ICT Centre, 2006.
- [132] Cecile Paris, Andrew Lampert, S Lu, and M Wu. Enhancing dynamic knowledge management services - tailored documents. Technical report, CSIRO ICT Centre, 2005.
- [133] Cécile Paris, Stephen Wan, Ross Wilkinson, and Mingfang Wu. Generating personal travel guides - and who wants them? In *UM '01: Proceedings of the 8th International Conference on User Modeling 2001*, pages 251–253, London, UK, 2001. Springer-Verlag.
- [134] Cécile L. Paris. Tailoring object descriptions to a user's level of expertise. *Comput. Linguist.*, 14(3):64–78, 1988.
- [135] Marius Pasca and Benjamin Van Durme. What you seek is what you get: Extraction of class attributes from query logs. In *IJCAI*, pages 2832–2837, 2007.
- [136] Marius Pasca and Benjamin Van Durme. Weakly-supervised acquisition of open-domain classes and class attributes from web documents and query logs. In *ACL*, pages 19–27, 2008.
- [137] Karen Pinel-Sauvagnat, Mohand Boughanem, and Claude Chrisment. Answering content-and-structure-based queries on XML documents using relevance propagation. *Information Systems, Special Issue SPIRE 2004*, 31:621–635, janvier 2006. Impact Factor: 1,887.

- [138] Karen Pinel-Sauvagnat, Mohand Boughanem, and Claude Christment. Why using structural hints in XML retrieval ? (regular paper). In Henrik Legind Larsen, Gabriella Pasi, and Ortiz-Arroyo Daniel, editors, *Flexible Query Answering (FQAS), Milan, Italie, 07/06/2006-10/06/2006*, Advances in Artificial Intelligence, pages 197–109, <http://www.worldscientific.com>, juin 2006. World Scientific. Taux d’acceptation non communiqué.
- [139] Ashok Kumar Ponnuswami, Kumaresh Pattabiraman, Qiang Wu, Ran Gilad-Bachrach, and Tapas Kanungo. On composition of a federated web search result page: using online users to provide pairwise preference for heterogeneous verticals. In *Proceedings of the fourth ACM international conference on Web search and data mining, WSDM ’11*, pages 715–724, New York, NY, USA, 2011. ACM.
- [140] Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR ’98*, pages 275–281, New York, NY, USA, 1998. ACM.
- [141] Ana-Maria Popescu and Oren Etzioni. Extracting product features and opinions from reviews. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing, HLT ’05*, pages 339–346, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.
- [142] M. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.
- [143] Long Qiu, Min-Yen Kan, and Tat-Seng Chua. Paraphrase recognition via dissimilarity significance classification. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 18–26, Sydney, Australia, July 2006. Association for Computational Linguistics.
- [144] Anand Ranganathan, Anton Riabov, and Octavian Udrea. Mashup-based information retrieval for domain experts. In *CIKM ’09: Proceeding of the 18th ACM conference on Information and knowledge management*, pages 711–720, New York, NY, USA, 2009. ACM.
- [145] Jane Reid, Mounia Lalmas, Karen Finesilver, and Morten Hertzum. Best entry points for structured document retrieval: part i: characteristics. *Inf. Process. Manage.*, 42(1):74–88, 2006.
- [146] Stephen E. Robertson. The probability ranking principle in ir. *Journal of documentation*, 1977.

- [147] Stephen E. Robertson and Steve Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In W. Bruce Croft and C. J. van Rijsbergen, editors, *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval. Dublin, Ireland, 3-6 July 1994 (Special Issue of the SIGIR Forum)*, pages 232–241. ACM/Springer, 1994.
- [148] Cyril Rohr and Dian Tjondronegoro. Aggregated cross-media news visualization and personalization. In *MIR '08: Proceeding of the 1st ACM international conference on Multimedia information retrieval*, pages 371–378, New York, NY, USA, 2008. ACM.
- [149] Nachiketa Sahoo, Jamie Callan, Ramayya Krishnan, George Duncan, and Rema Padman. Incremental hierarchical clustering of text documents. In *CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 357–366, New York, NY, USA, 2006. ACM.
- [150] G. Salton, J. Allan, and C. Buckley. Approaches to Passage Retrieval in Full Text Information Systems. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 49–58, 1993.
- [151] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18:613–620, November 1975.
- [152] Gerard Salton. *The SMART Retrieval System—Experiments in Automatic Document Processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1971.
- [153] Mark Sanderson and Janet Kohler. Analyzing geographic queries. In *Workshop on Geographic Information Retrieval*, 2005.
- [154] Rodrygo L. T. Santos, Craig Macdonald, and Iadh Ounis. Aggregated search result diversification. In *Proceedings of the 3rd International Conference on the Theory of Information Retrieval*, Bertinoro, Italy, 2011. Springer.
- [155] Christina Sauper and Regina Barzilay. Automatically generating wikipedia articles: a structure-aware approach. In *ACL-IJCNLP 09: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1*, pages 208–216, Morristown, NJ, USA, 2009. Association for Computational Linguistics.

- [156] Erik Selberg and Oren Etzioni. Multi-service search and comparison using the metacrawler. In *In Proceedings of the 4th International World Wide Web Conference*, pages 195–208, 1995.
- [157] B. Settles. ABNER: An open source tool for automatically tagging genes, proteins, and other entity names in text. *Bioinformatics*, 21(14):3191–3192, 2005.
- [158] Sushmita Shanu, Lalmas Mounia, and Tombros Anastasio. Using digest pages to increase user result space: Preliminary design, 2008.
- [159] Michael Shilman. Aggregate documents: making sense of a patchwork of topical documents. In *DocEng '08: Proceeding of the eighth ACM symposium on Document engineering*, pages 3–7, New York, NY, USA, 2008. ACM.
- [160] Milad Shokouhi, Justin Zobel, Saied Tahaghoghi, and Falk Scholer. Using query logs to establish vocabularies in distributed information retrieval. *Inf. Process. Manage.*, 43:169–180, January 2007.
- [161] Karen Spärck-Jones, Stephen E. Robertson, and Mark Sanderson. Ambiguous requests: implications for retrieval tests, systems and theories. *SIGIR Forum*, 41(2):8–17, 2007.
- [162] K Srinivas, P V S Srinivas, and A Govardhan. A survey on the performance evaluation of various meta search engines. *International Journal of Computer Science Issues*, 8:359–364, 2011.
- [163] Andreas Strotmann Strotmann and Dangzhi Zhao. Bibliometric maps for aggregated visual browsing in digital libraries. In *SIGIR 2008 Workshop on aggregated search*.
- [164] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 697–706, New York, NY, USA, 2007. ACM.
- [165] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. YAGO: A large ontology from Wikipedia and Wordnet. *Web Semant.*, 6(3):203–217, 2008.
- [166] Shanu Sushmita, Hideo Joho, and Mounia Lalmas. A task-based evaluation of an aggregated search interface. In *SPIRE '09: Proceedings of the 16th International Symposium on String Processing and Information Retrieval*, pages 322–333, Berlin, Heidelberg, 2009. Springer-Verlag.

- [167] Shanu Sushmita, Hideo Joho, Mounia Lalmas, and Joemon M. Jose. Understanding domain relevance in web search. In *WWW 2009 Workshop on Web Search Result Summarization and Presentation, Madrid*, 2009.
- [168] Shanu Sushmita, Hideo Joho, Mounia Lalmas, and Robert Villa. Factors affecting click-through behavior in aggregated search interfaces. In *CIKM '10: Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 519–528, New York, NY, USA, 2010. ACM.
- [169] Zoltán Szilávik, Anastasios Tombros, and Mounia Lalmas. Feature- and query-based table of contents generation for xml documents. In *ECIR'07: Proceedings of the 29th European conference on IR research*, pages 456–467, Berlin, Heidelberg, 2007. Springer-Verlag.
- [170] Bilyana Taneva, Mouna Kacimi, and Gerhard Weikum. Gathering and ranking photos of named entities with high precision, high recall, and diversity. In *Proceedings of the third ACM international conference on Web search and data mining, WSDM '10*, pages 431–440, New York, NY, USA, 2010. ACM.
- [171] R. S. Taylor. The process of asking questions. *American Documentation*, 13(4):391–396, 1962.
- [172] Taro Tezuka and Katsumi Tanaka. Temporal and spatial attribute extraction from web documents and time-specific regional web search system. In *W2GIS*, pages 14–25, 2004.
- [173] Paul Thomas, Katherine Noack, and Cecile Paris. Evaluating interfaces for government metasearch. In *Proceeding of the third symposium on Information interaction in context, IiX '10*, pages 65–74, New York, NY, USA, 2010. ACM.
- [174] Kosuke Tokunaga and Kentaro Torisawa. Automatic discovery of attribute words from web documents. In *In Proceedings of the 2nd International Joint Conference on Natural Language Processing (IJCNLP-05)*, pages 106–118, Jeju Island, Korea, pages 106–118, 2005.
- [175] Andrew Trotman, Shlomo Geva, Jaap Kamps, Mounia Lalmas, and Vanessa Murdock. Current research in focused retrieval and result aggregation. *Journal of Information Retrieval*, 2010.
- [176] Peter D. Turney. Mining the Web for synonyms: PMI-IR versus lsa on toefl. In *EMCL '01: Proceedings of the 12th European Conference on Machine Learning*, pages 491–502, London, UK, 2001. Springer-Verlag.

- [177] Subodh Vaid, Christopher B. Jones, Hideo Joho, and Mark Sanderson. Spatio-textual indexing for geographical search on the web. In *SSTD*, pages 218–235, 2005.
- [178] David Vallet and Hugo Zaragoza. Inferring the most important types of a query: a semantic approach. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 857–858, New York, NY, USA, 2008. ACM.
- [179] Anne-Marie Vercoustre, Jovan Pehcevski, and James A. Thom. Focused access to xml documents. chapter Using Wikipedia Categories and Links in Entity Ranking, pages 321–335. Springer-Verlag, Berlin, Heidelberg, 2008.
- [180] Ellen M. Voorhees. Evaluating answers to definition questions. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 109–111, Morristown, NJ, USA, 2003. Association for Computational Linguistics.
- [181] Ellen M. Voorhees, Narendra K. Gupta, and Ben Johnson-Laird. Learning collection fusion strategies. In *SIGIR '95: Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 172–179, New York, NY, USA, 1995. ACM.
- [182] Courtney Wade and James Allan. Passage retrieval and evaluation. Technical report, 2005.
- [183] Chuang Wang, Xing Xie, Lee Wang, Yansheng Lu, and Wei-Ying Ma. Detecting geographic locations from web resources. In *GIR '05: Proceedings of the 2005 workshop on Geographic information retrieval*, pages 17–24, New York, NY, USA, 2005. ACM.
- [184] Richard C. Wang and William W. Cohen. Iterative set expansion of named entities using the web. In *ICDM '08: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, pages 1091–1096, Washington, DC, USA, 2008. IEEE Computer Society.
- [185] Ian H. Witten and Eibe Frank. Data mining: practical machine learning tools and techniques with java implementations. *SIGMOD Rec.*, 31:76–77, March 2002.
- [186] Tak-Lam Wong and Wai Lam. A probabilistic approach for adapting information extraction wrappers and discovering new attributes. In *Proceedings of the Fourth IEEE International Conference on Data*



- Mining*, ICDM '04, pages 257–264, Washington, DC, USA, 2004. IEEE Computer Society.
- [187] Tak-Lam Wong and Wai Lam. An unsupervised method for joint information extraction and feature mining across different web sites. *Data Knowl. Eng.*, 68:107–125, January 2009.
- [188] Fei Wu, Raphael Hoffmann, and Daniel S. Weld. Information extraction from wikipedia: moving down the long tail. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 731–739, New York, NY, USA, 2008. ACM.
- [189] M. Wu and M. Fuller. Supporting the answering process. In *In Proceedings of the Second Australian Document Computing Symposium*, pages 65–73, 1997.
- [190] Mingfang Wu, Michael Fuller, and Ross Wilkinson. Using clustering and classification approaches in interactive retrieval. *Inf. Process. Manage.*, 37(3):459–484, 2001.
- [191] Shengli Wu and Fabio Crestani. Data fusion with estimated weights. In *CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management*, pages 648–651, New York, NY, USA, 2002. ACM.
- [192] Shengli Wu and Sally McClean. Performance prediction of data fusion for information retrieval. *Inf. Process. Manage.*, 42(4):899–915, 2006.
- [193] Minoru Yoshida and Kentaro Torisawa. A method to integrate tables of the world wide web. In *In Proceedings of the International Workshop on Web Document Analysis (WDA 2001)*, pages 31–34, 2001.
- [194] Naoki Yoshinaga and Kentaro Torisawa. Open-domain attribute-value acquisition from semi-structured texts. In *Proceedings of the workshop on Ontolex*, pages 55–66, 2007.
- [195] Gae-won You, Seung-won Hwang, Zaiqing Nie, and Ji-Rong Wen. Socialsearch: enhancing entity search with social network matching. In *Proceedings of the 14th International Conference on Extending Database Technology, EDBT/ICDT '11*, pages 515–519, New York, NY, USA, 2011. ACM.
- [196] Ke Zhou, Ronan Cummins, and Mounia Lalmas. Evaluating large-scale distributed vertical search. In *The 9th International Workshop on Large-Scale and Distributed Systems for Information Retrieval, LS-DSIR '11*, page to appear. ACM, 2011.